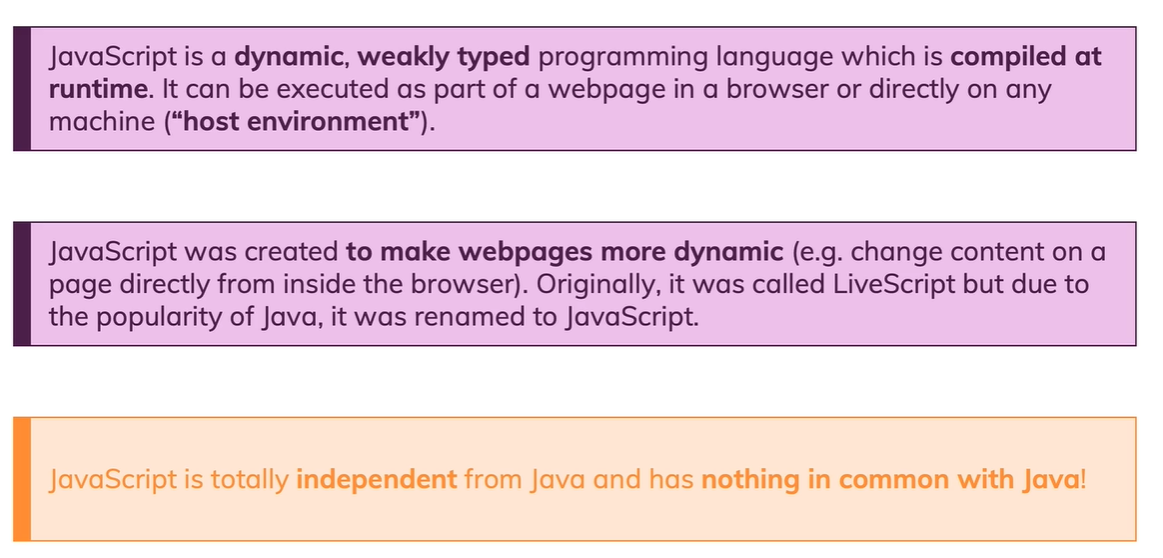
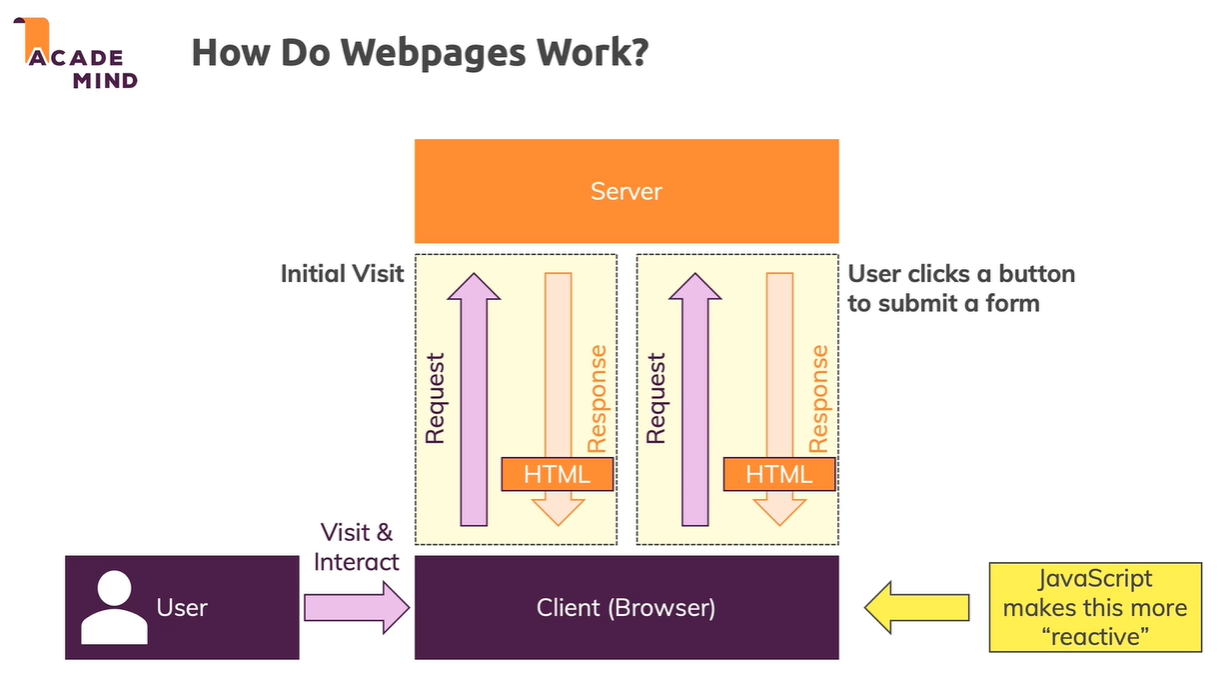
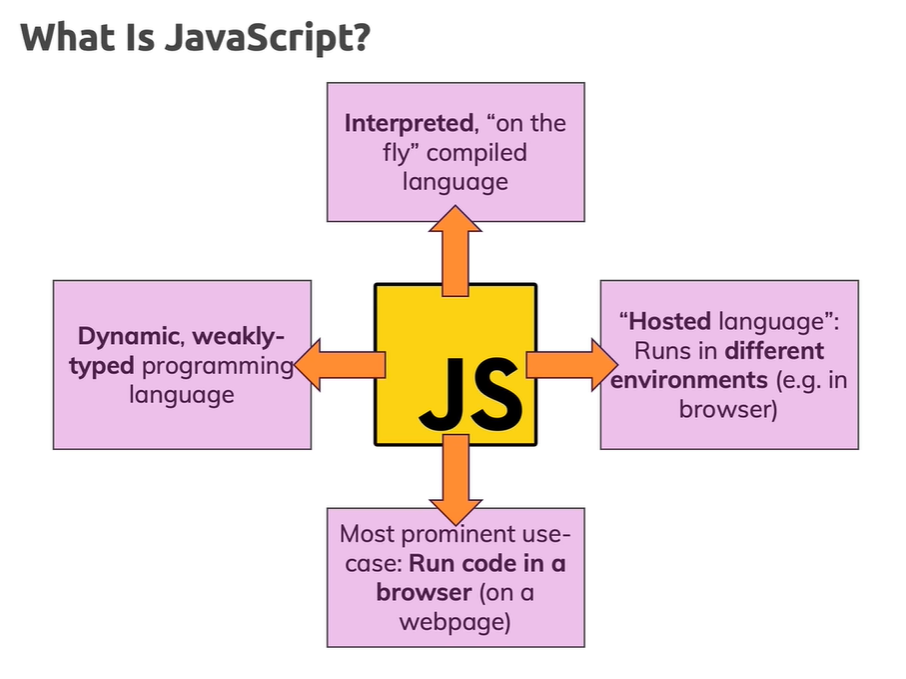
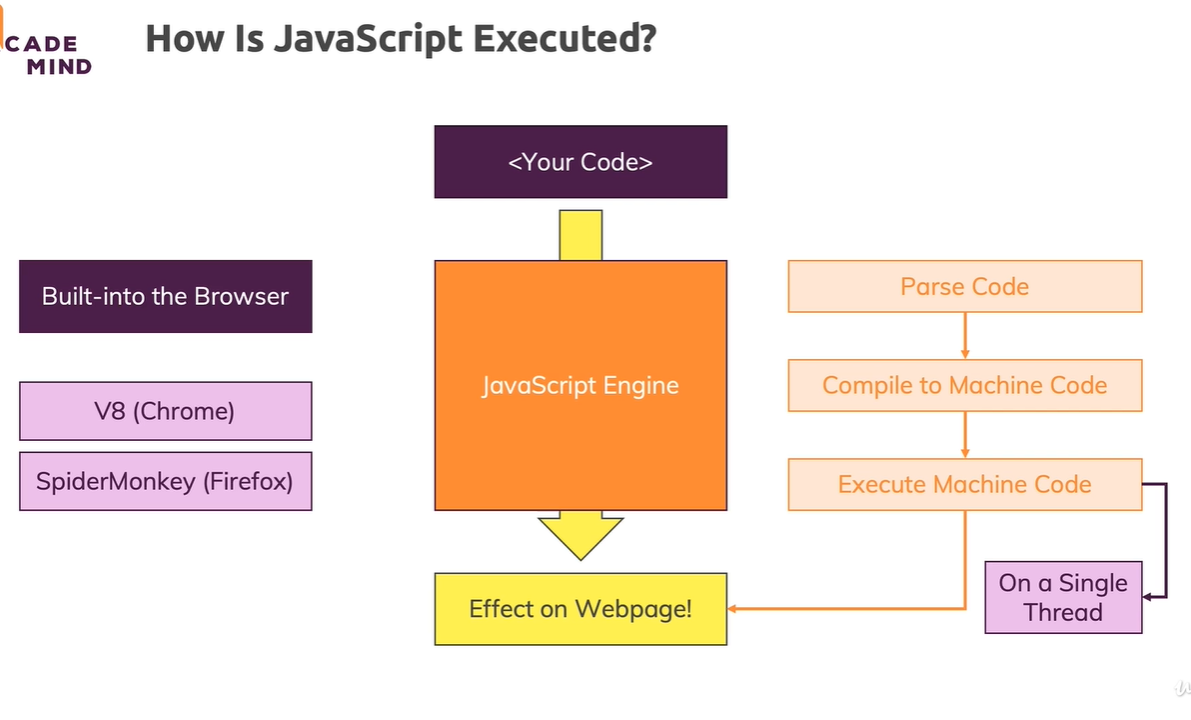
**JavaScript**

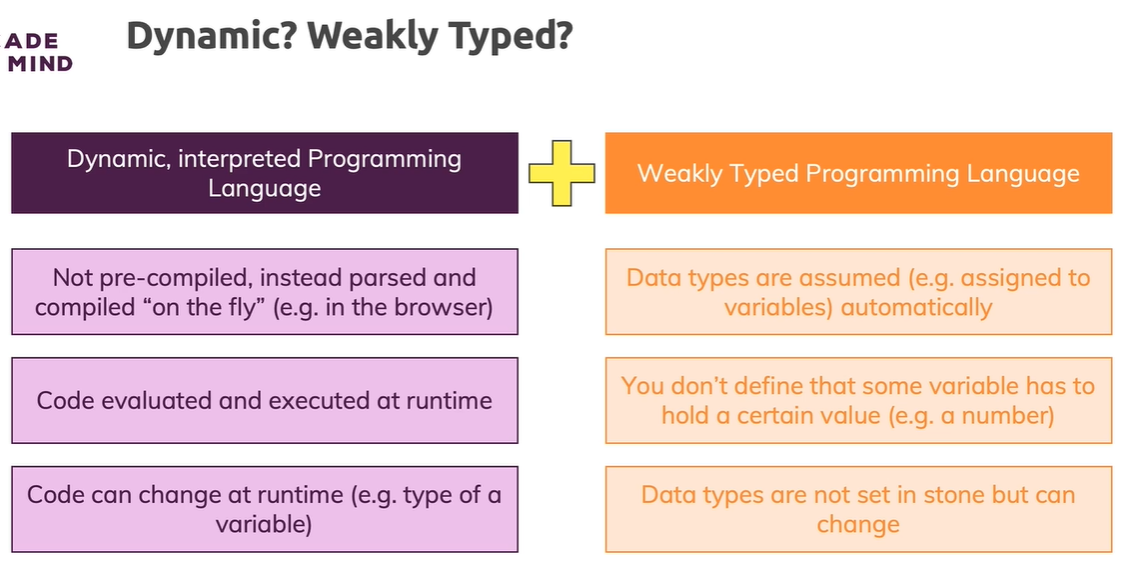
1. Introduction

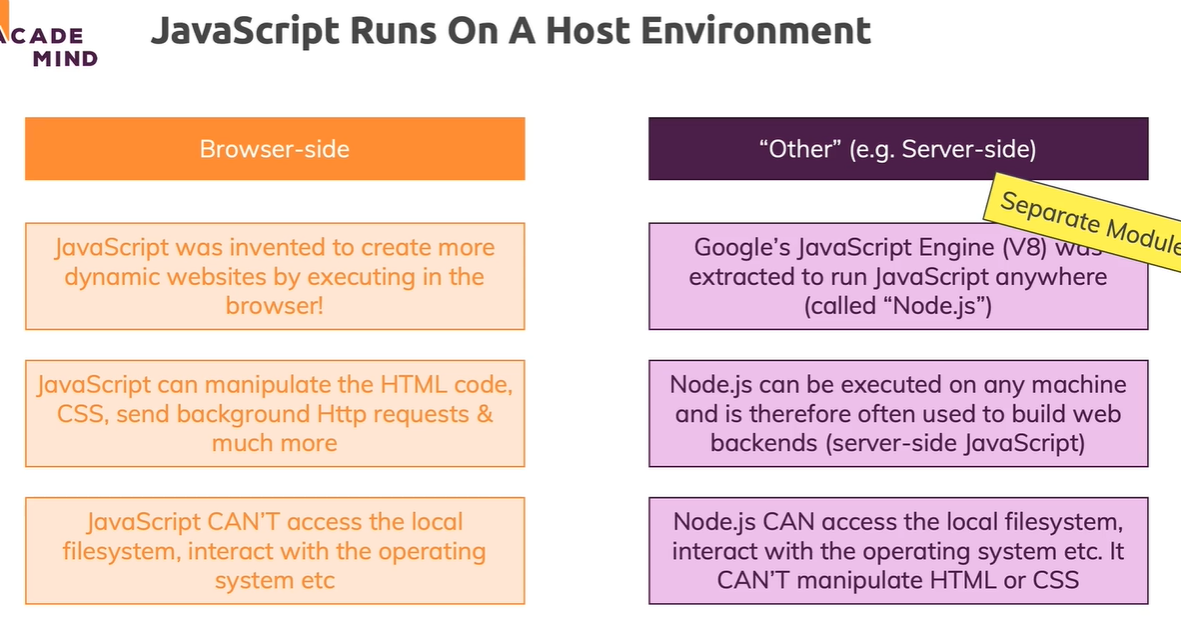


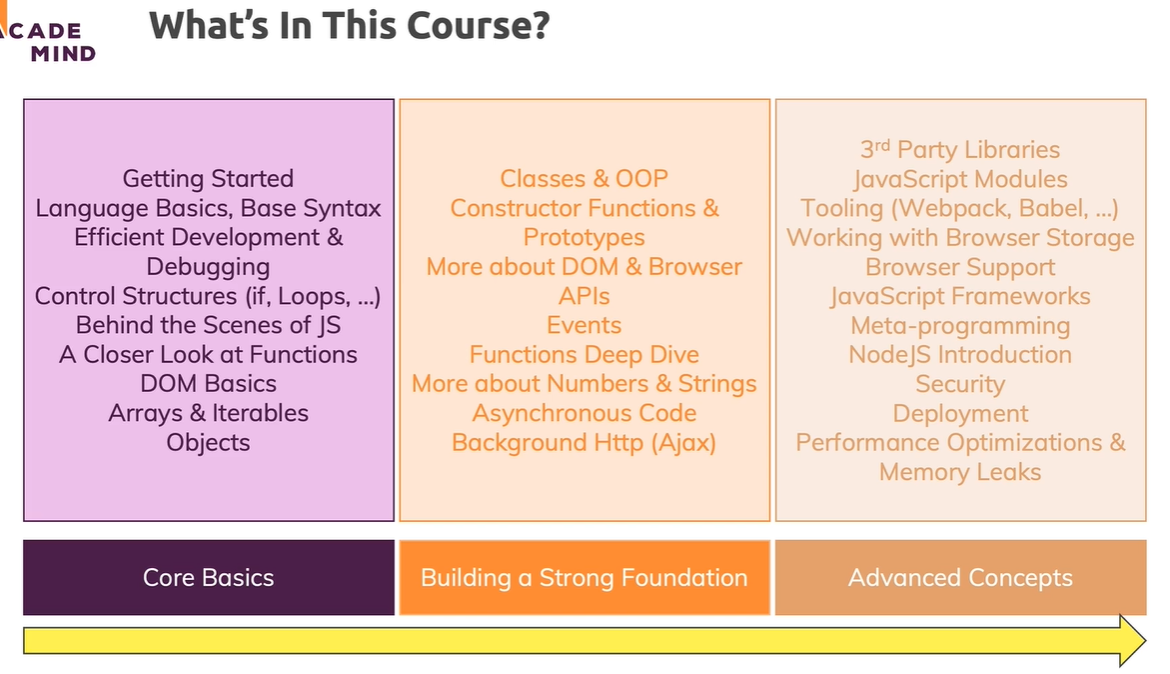


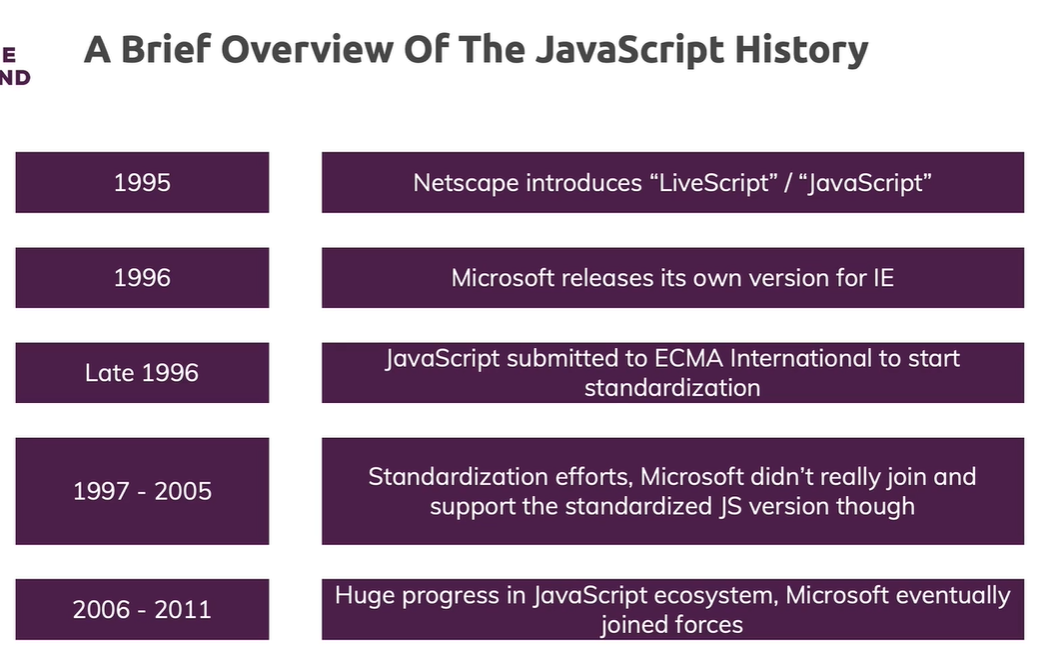


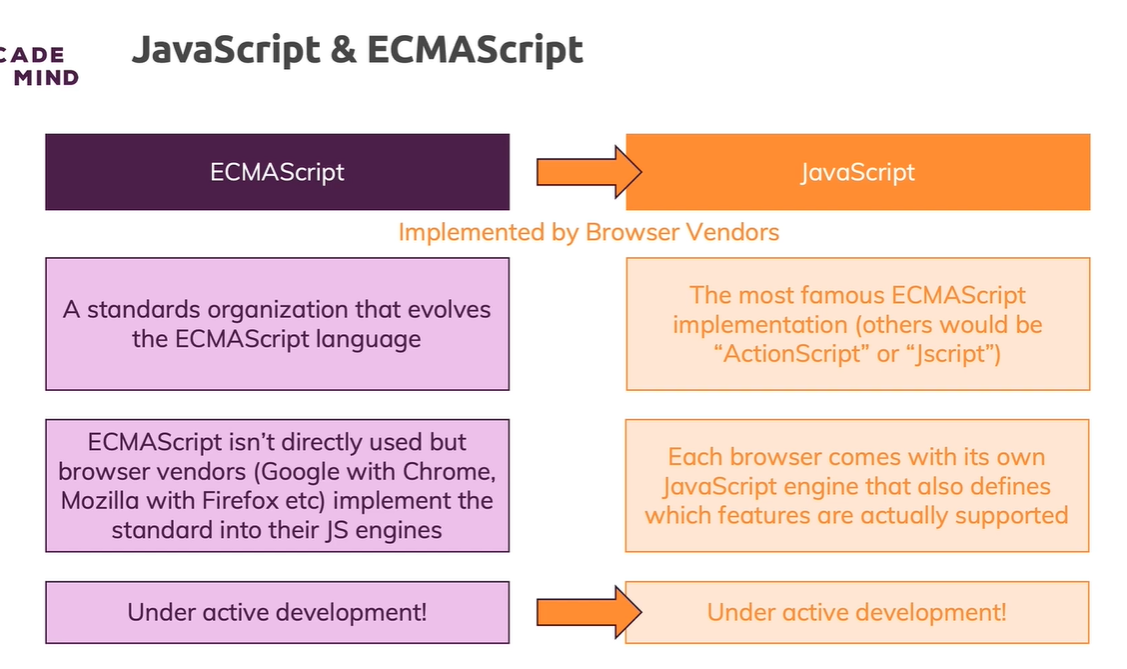




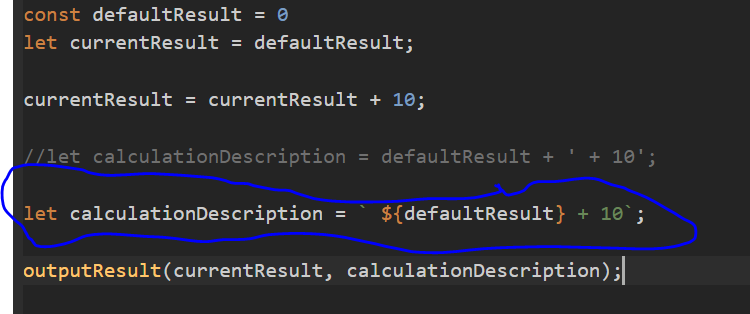








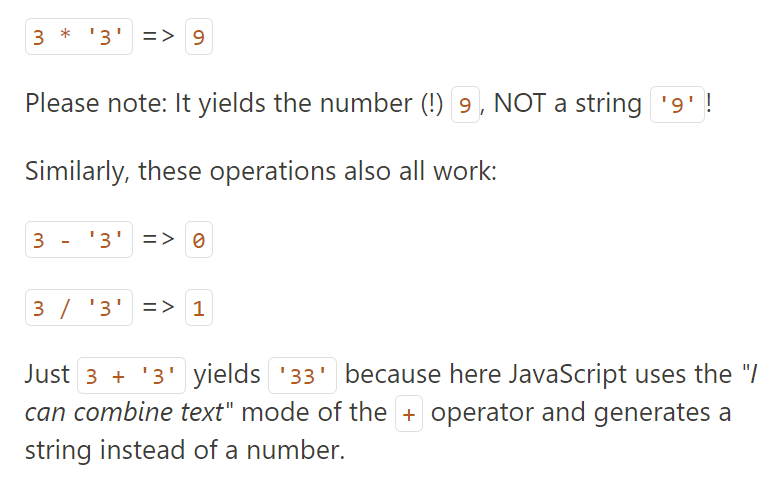
1. Basics: Variables, Data Types, Operators & Functions
   1. **Variables** (let) and **Constants** (const)
      1. Use more constant whenever it looks data will not change in any where
      2. Variable naming
         1. Best practice: camelCase
         2. Only letters and digits
         3. Starting with $ is allowed
         4. Starting with \_ is allowed
   2. Semicolon is optional in Java script (but some case you must enter ;)
   3. **Exponentiation Operator** in Java script (2\*\*3 = 8)
   4. Data Types
      1. Numbers (2, -3, 22.987)
      2. String (‘Hi’, “Hi”, `Hi`)
      3. Boolean (true, false)
      4. Objects {name: ‘Max’, age: 31}
      5. Arrays [1,2,3]
   5. Backtick Symbol `${variable}



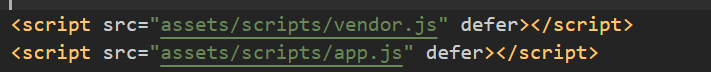
* 1. Show whitespace (line break) between text
     1. Browser default remove white between text
     2. To show whitespace use CSS code **{ white-space: pre}**
  2. Introducing Functions
     1. Code on demand
        1. Define function
        2. Call function
     2. In case of variable, it should declare first then access them (order matter). But function we can define anywhere and access from anywhere (order don’t matter).
  3. Global & Local Scope
     1. **Pure function**
        1. If a function which will not change any global variable. Only work on local or argument variable.
     2. Shadowed Variables
        1. If we have a variable with same name in both global and local scope, then It creates a new variable on a different scope - this variable does not overwrite or remove the global variable by the way - both co-exist.



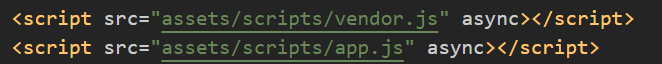
* 1. "Indirect" vs "Direct" Function Execution
     1. Direct – add (1, 2)
     2. Indirect - addBtn.addEventListener("click", add);
        1. Here we pass only function and tell browser that when we click then execute add function.
  2. Converting Data Types
     1. String into number
        1. **parseInt**() {e.g. parseInt(variable)}
        2. **+ operator** {e.g. +variable}
     2. Number into String
        1. **toString**() {e.g variable.toString()}



* 1. Comments
     1. Single line - // comment
     2. Multiple line - /\* comment \*/
  2. Undefined, null & NaN
     1. These are special values
     2. undefined
        1. Default value of uninitialized variables
        2. You shouldn’t assign undefined as a value manually
     3. null
        1. We can use this to assign as value to reset variables
     4. NaN
        1. It’s a type of number and can be used in calculations
        2. E.g. 3 \* ‘Hi’ = NaN
  3. The typeof operator
     1. Return type of variable or constant
     2. E.g typeof ‘Ankit’ = ‘string’, typeof 1= ‘number’
     3. Array is type of object, e.g. typeof [1,2,3] = ‘object’
     4. E.g. typeof undefined = “undefined”, typeof null = “object”, typeof NaN = “number”
  4. Importing Script correctly
     1. Import script file in bottom of body tag in html files.
     2. **defer** attribute in script tag ensures that scripts gets loaded early but executedonly once parsing HTML finished

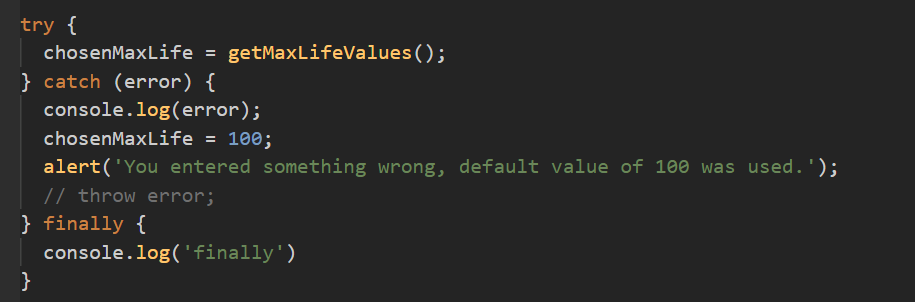


* + 1. **async** allow to script really executes as early as possible, so as soon as it was downloaded. The order of the script execution is therefore not guaranteed.

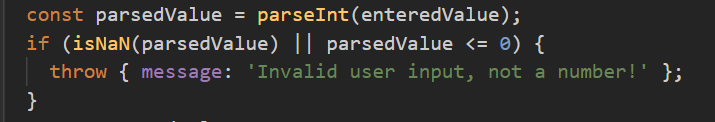


* + 1. Best way to import scripts in html either import in bottom of body tag or use defer if imported in head tag.

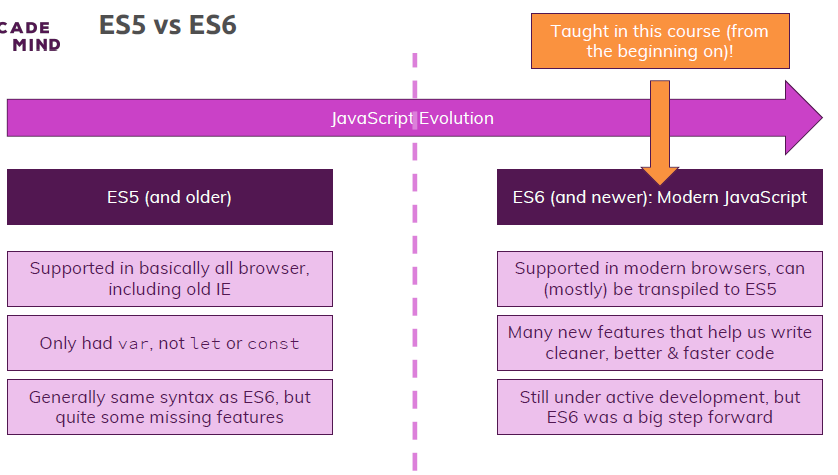
1. Debugging and Efficient Developing
   1. MDN site is good for java script tutorial
   2. Performance tab in chrome browser is used to check whole file loading process
2. Control Structures
   1. Conditions (If) statements
      1. If statement, if else statement, else if statement
   2. Falsy or Truthy values
      1. 0, empty string, null, undefined, NaN 🡺 false
      2. +ve or -ve value, non-empty string, object , array 🡺 true
   3. Logical Operator tricks
      1. !!”” = false, !!1 = true (use to convert truthy or falsy value into boolean)
      2. var a = b || ‘1’; (use to set with default value)
   4. Loops
      1. Execute same code multiple times
      2. for loop : execute code a certain amount of times (with counter variable)
      3. for-of loop: execute for every element in an array
      4. for-in loop: execute for every key in an object
      5. while loop or do-while loop: execute code as long as a condition is true
   5. Error handling
      1. try, catch and finally block



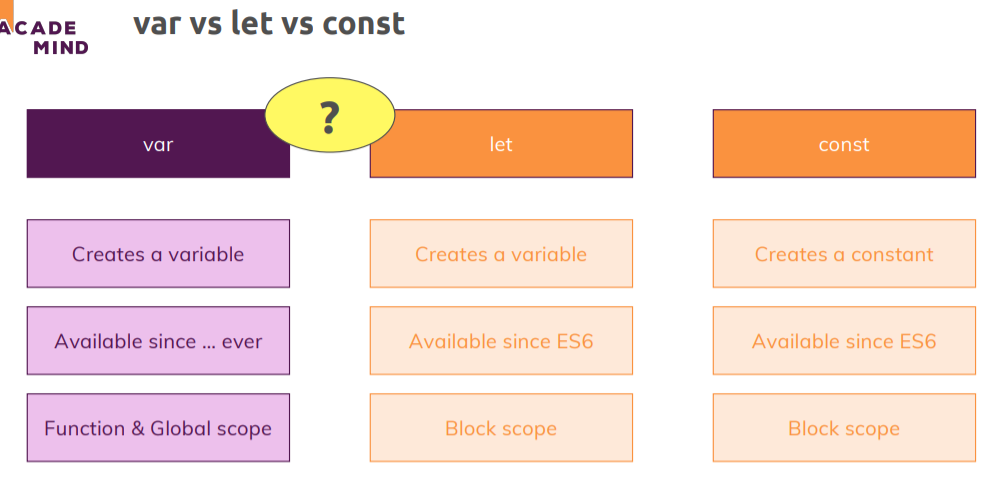
* + 1. throw custom error



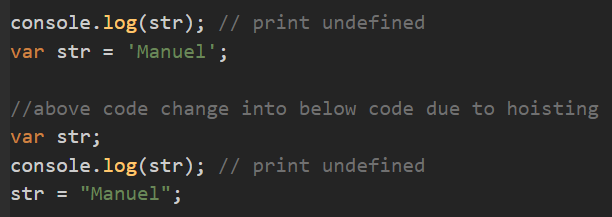
1. How it works, The Weird Parts & ES5 vs ES6+
   1. ES5 vs ES6+ (Next Gen Js, ECMAScript)



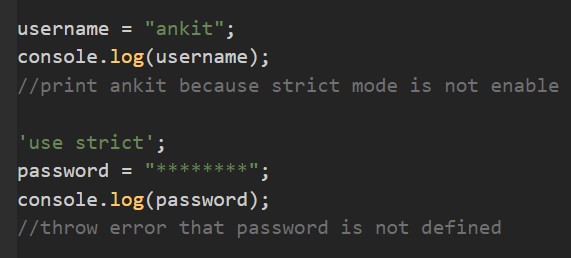
* 1. var vs let vs const



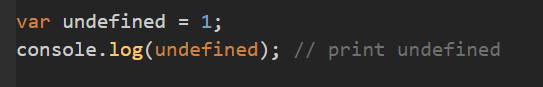
* 1. Hoisting
     1. If we use var variable before declare it then it print as undefined because due to hoisting in javascript all var declare move to top of the code.
     2. But this won’t happen in case of let or const so it throw error that variable can’t access before initialization.



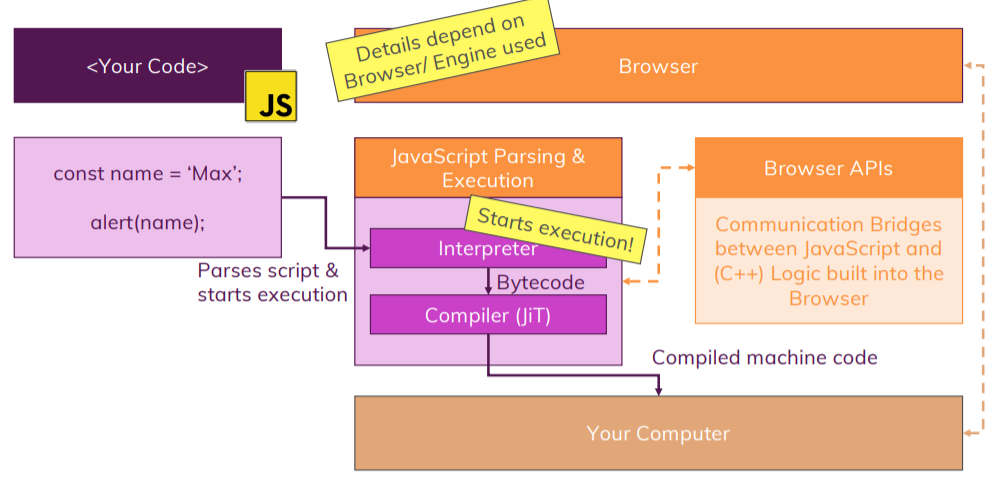
* 1. Strict Mode
     1. ‘use strict’; add at top in js file to enable strict mode
     2. 1st Case: if we declare variable without add var before it then It won’t throw error.



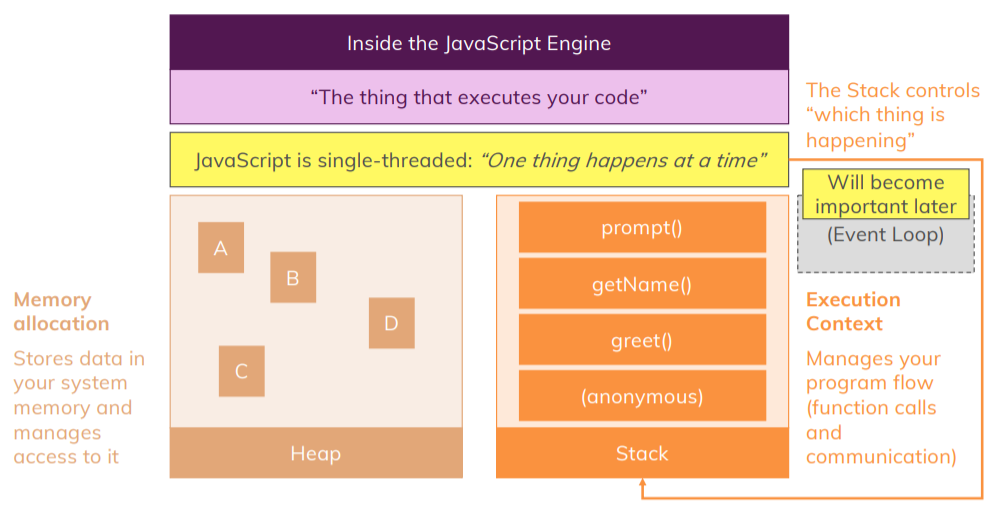
* + 1. 2nd Case: if we use reserved keyword as identifier with var (in case of let or const throw error) it won’t throw error.



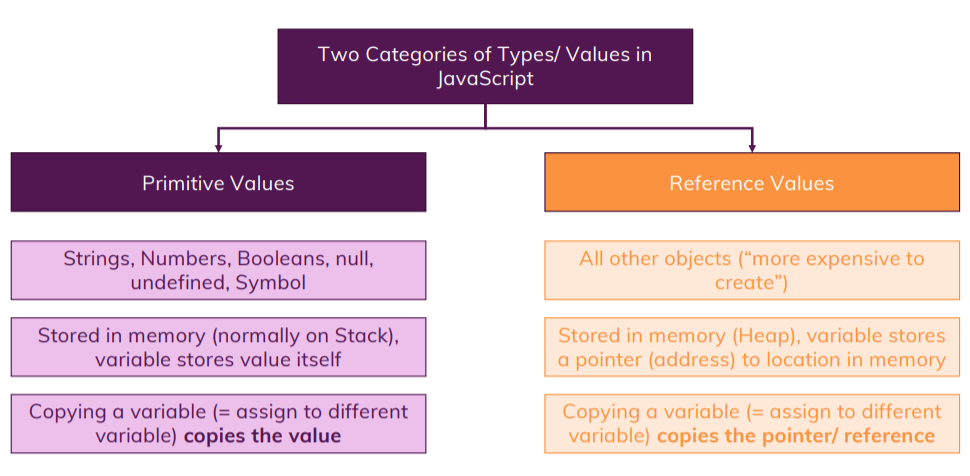
* 1. How code is parsed & compiled



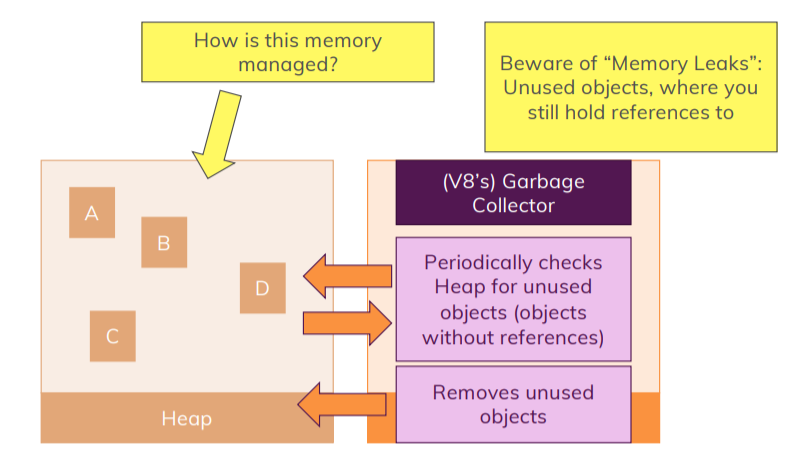
* 1. How code Gets Executed



* 1. Primitive vs Reference Values
     1. If we create const variable as object or array then we can change value but if we do assignment then only it throw error.



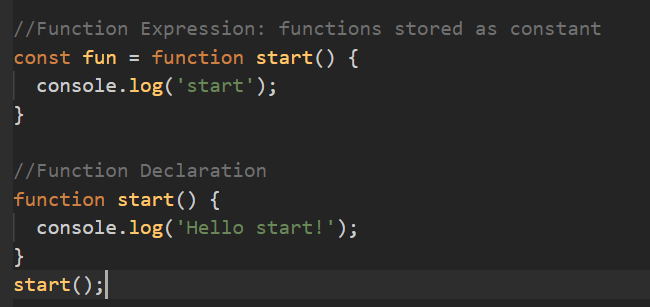
* 1. Garbage Collection



1. More about Functions
   1. Parameters vs Arguments
      1. Parameters are these variables which you specify between parentheses when defining a function.
      2. Arguments then are the concrete values you pass to a function when calling that function.
   2. Methods vs Functions



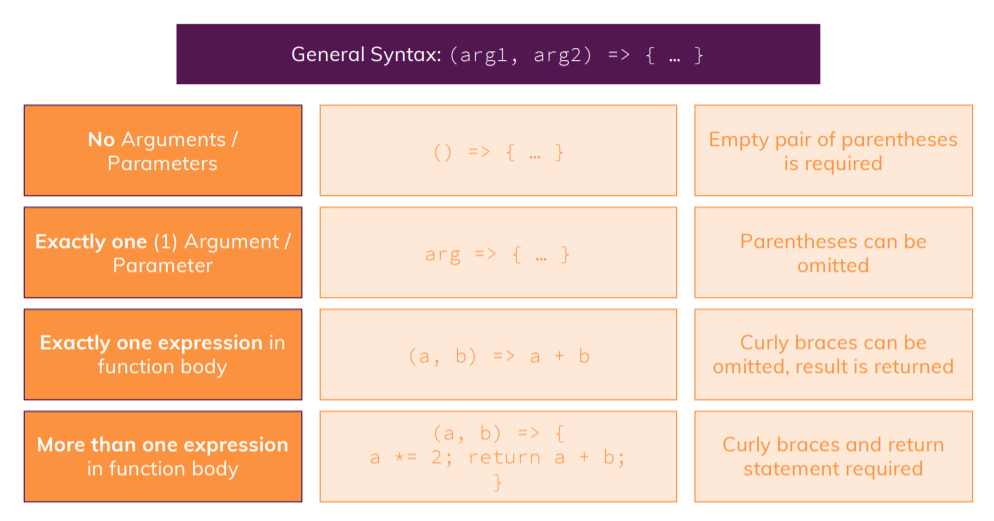
* 1. Functions
     1. Functions are objects, store as variable
     2. Functions can be stored in variable or constant (called Function Expression)
     3. If we stored function in variable and then called that function below declaring then it throw error. But in case for function declaration it work fine due to Hoisting it move in the top of file.



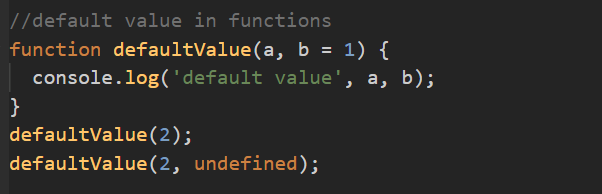
* + 1. Anonymous function



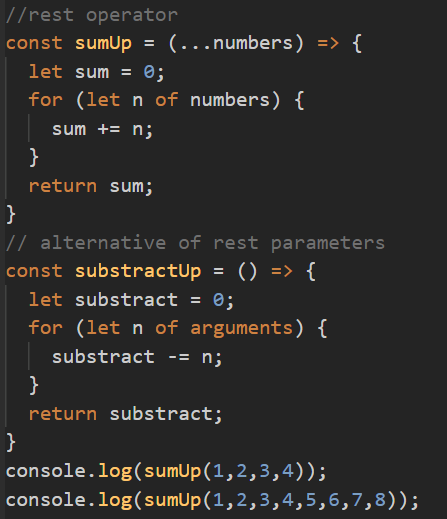
* + 1. Arrow Functions



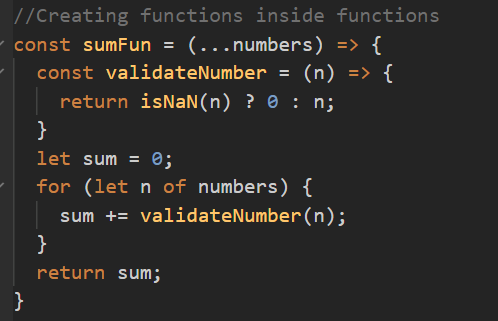
* + 1. Default Value in functions
       1. If we don’t pass argument in function call that means function take that parameter as undefined then if you set default value the it uses default value.



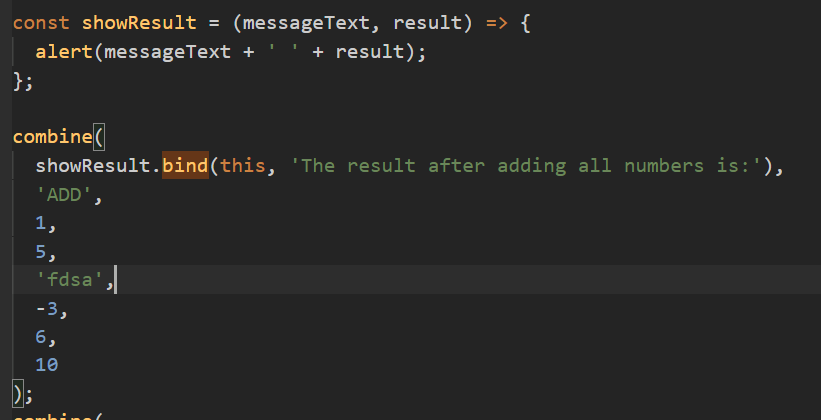
* + 1. Rest Parameters (Rest operator)
       1. Using rest operator we can accept infinite no of parameters and function make array of all parameter.
       2. Rest Parameters should be last parameter.
       3. Only one rest parameters we can accept in function parameter.
       4. Alternative of rest parameters is arguments (don’t use not best practice).



* + 1. Creating Functions inside Functions

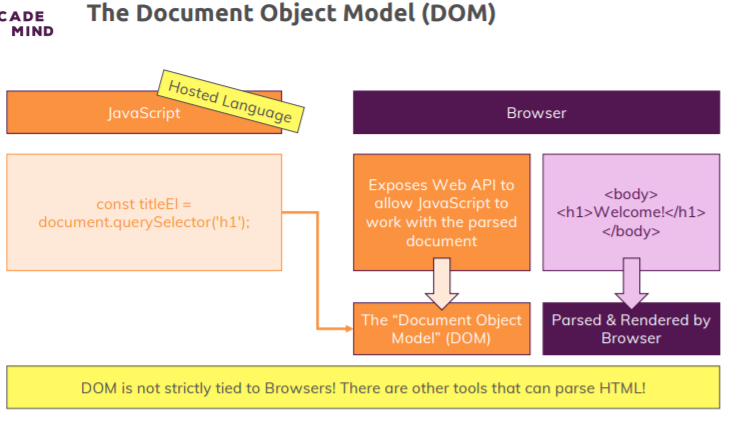


* + 1. Callback function
       1. If we pass function as argument then its called callback as it will execute not that time but later.
    2. Working with bind()

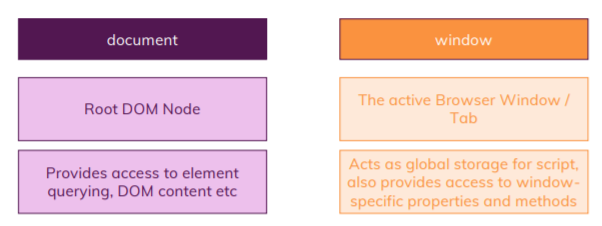


* + 1. call() vs apply()

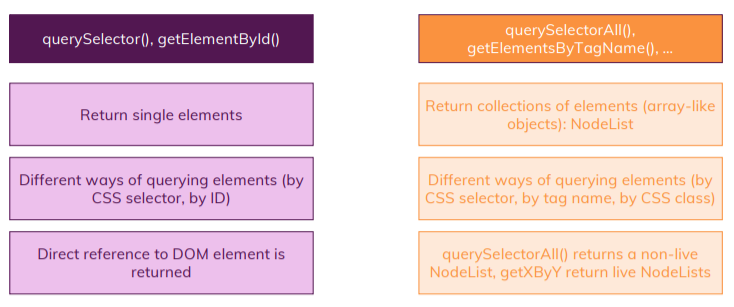
1. Exploring with DOM
   1. What is DOM



* 1. Document vs Window



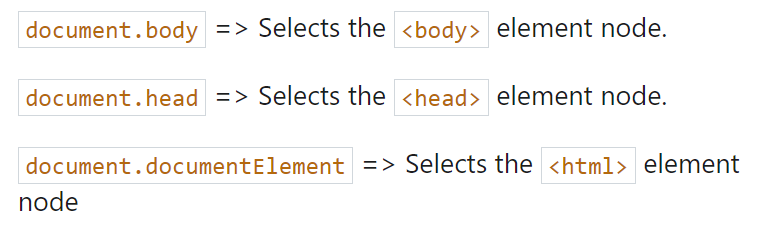
* 1. Querying elements



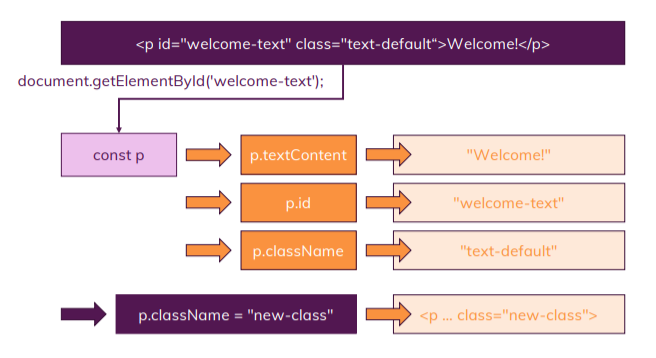
* 1. Nodes and Elements



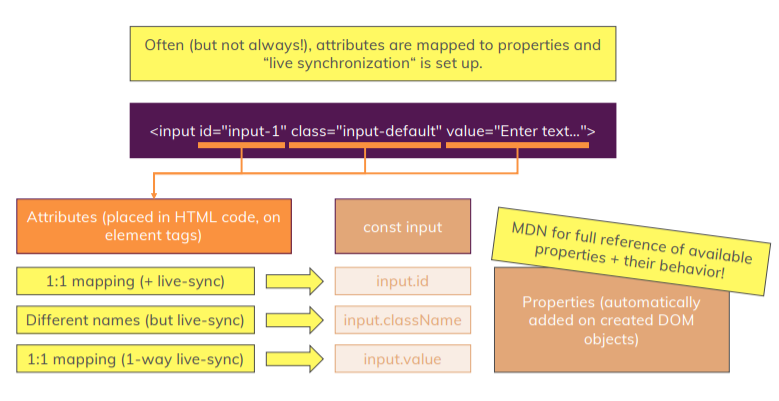
* 1. Selecting Elements in the DOM



* 1. Evaluating & Manipulating elements

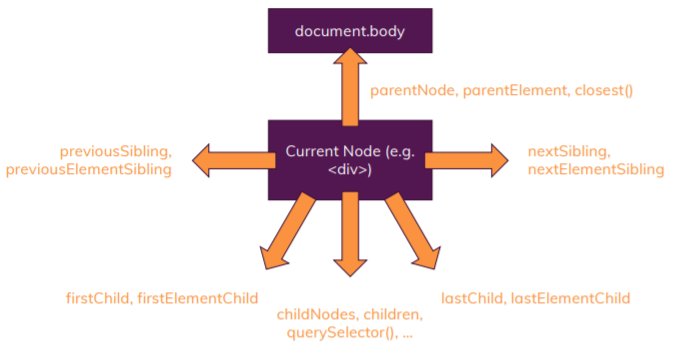


* 1. Attributes vs Properties

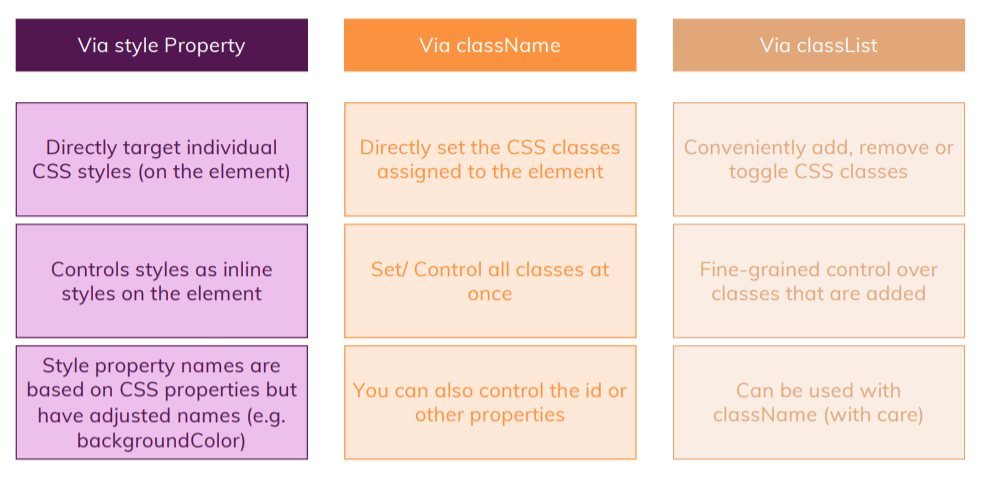


* 1. Traversing the DOM

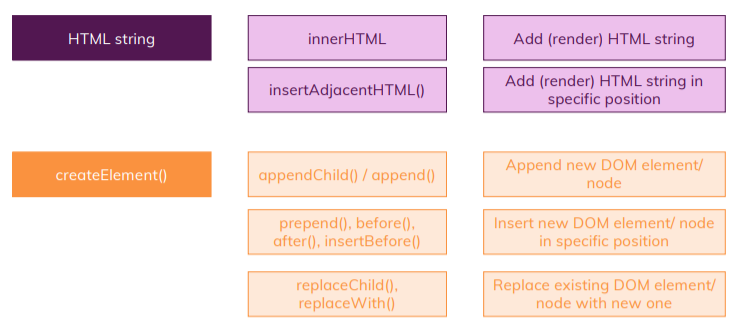




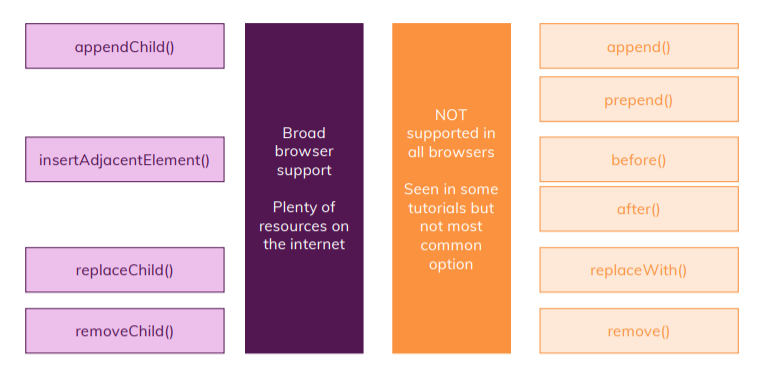
* 1. Styling DOM elements



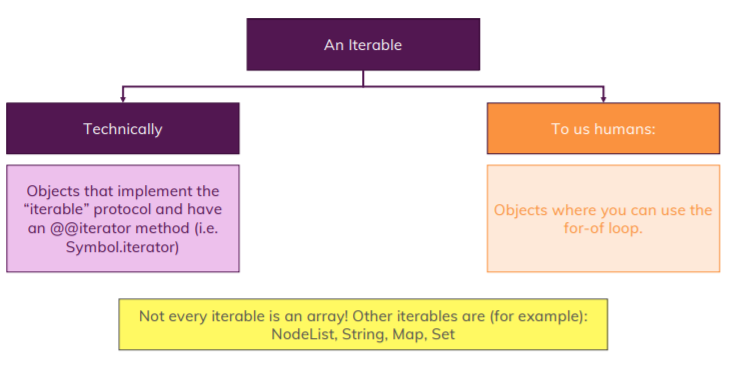
* 1. Creating & Inserting Elements



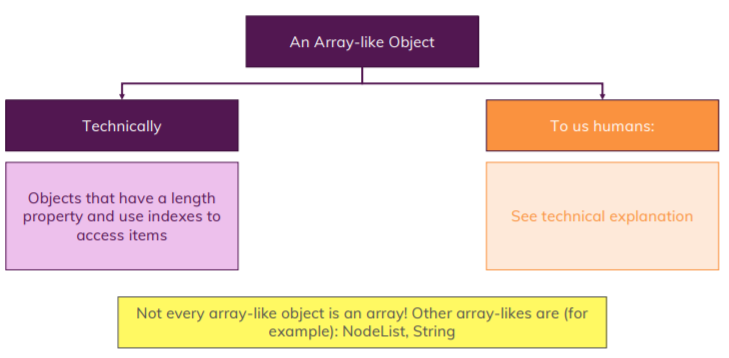
* 1. Insertion & Removal Elements



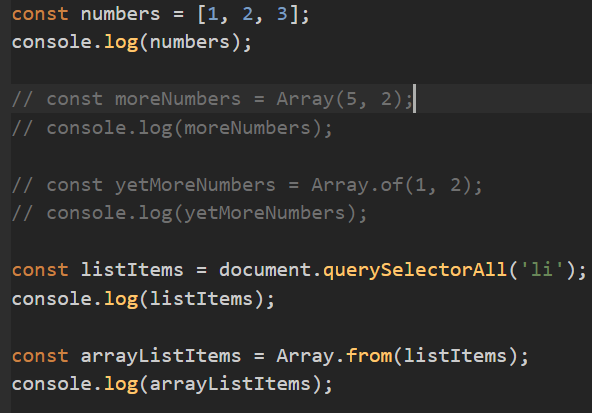
1. Arrays & Iterables
   1. What is an Iterable?



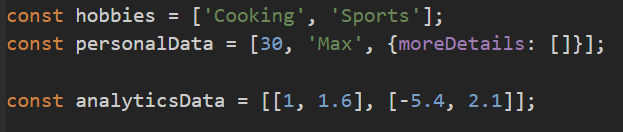
* 1. What‘s an “Array-like Object”?



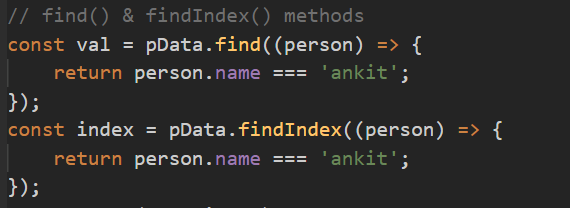
* 1. Creating Arrays



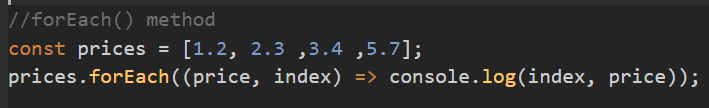
* 1. Which data can be store in Array?



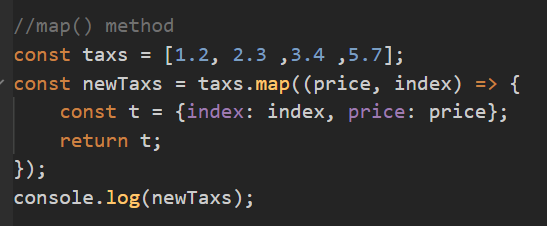
* 1. Adding & Removing elements
     1. The push(): add new value in last
     2. The unshift(): add new value in first
     3. The pop(): remove last value
     4. The shift(): remove first value
     5. The splice(index, count, …newValues): remove value from index and count
     6. The slice(start, end): return copy of array from start to end.
     7. The concat(…numbers): add array in end of existing array
     8. The indexOf(): return value of index from start
     9. The lastIndexOf(): return value of index from end
     10. The includes(): Determines whether an array includes a certain element, returning true or false as appropriate.



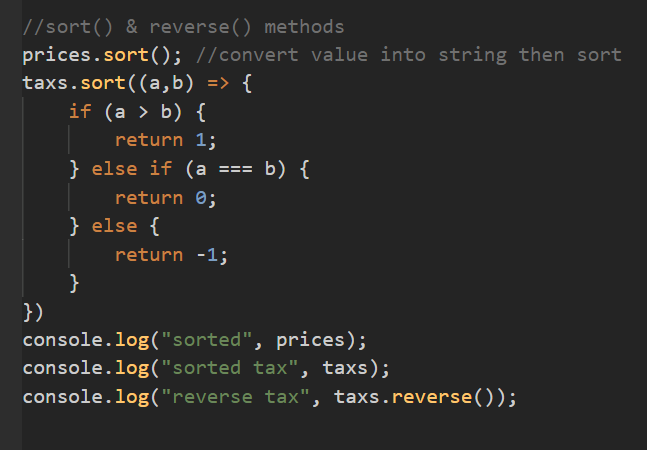
* + 1. Alternative to for loops: forEach() method



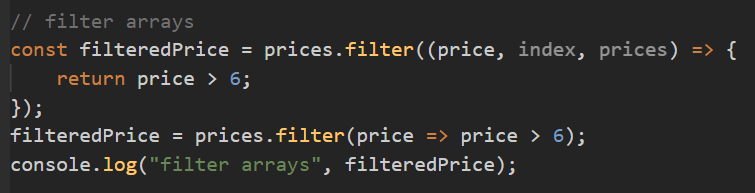
* + 1. Transforming data with map()



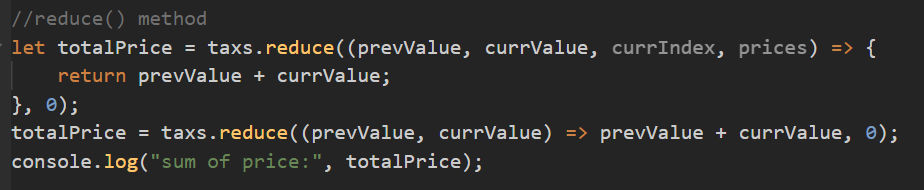
* + 1. The sort() and reverse() method
       1. Sort by default covert value into string and then sort it basis of first character.



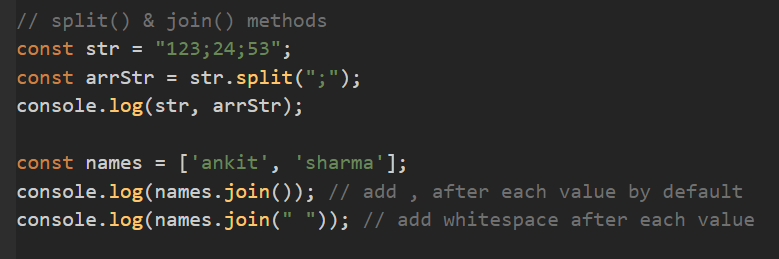
* + 1. Filtering arrays with filter()
       1. Filter array based on condition and return new array



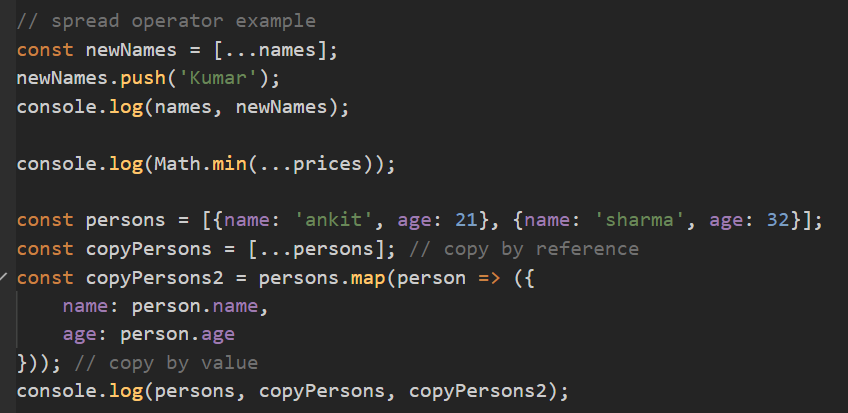
* + 1. The reduce() method



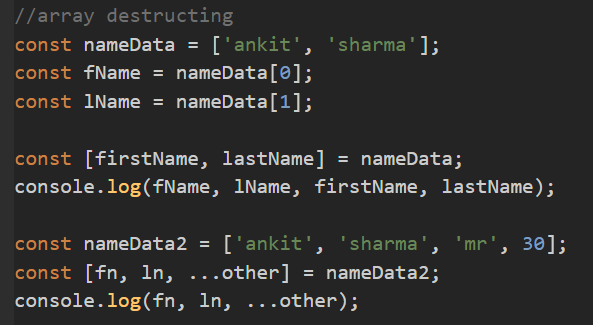
* + 1. The split() and join() method



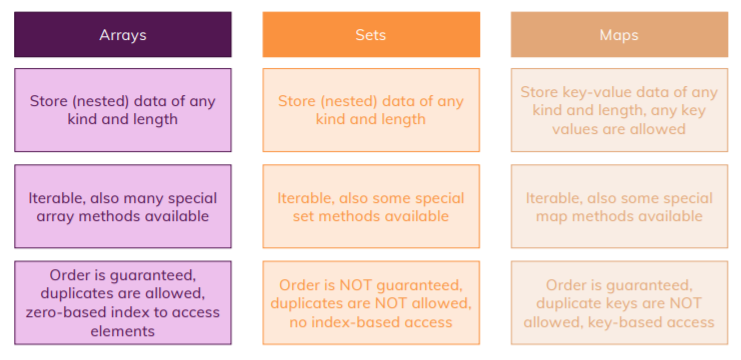
* + 1. The Spread Operator […]



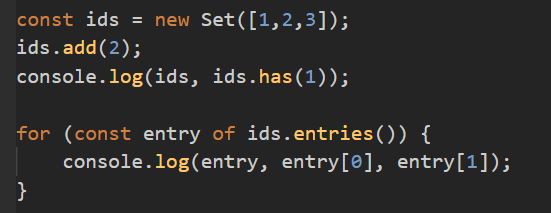
* + 1. Array destructing



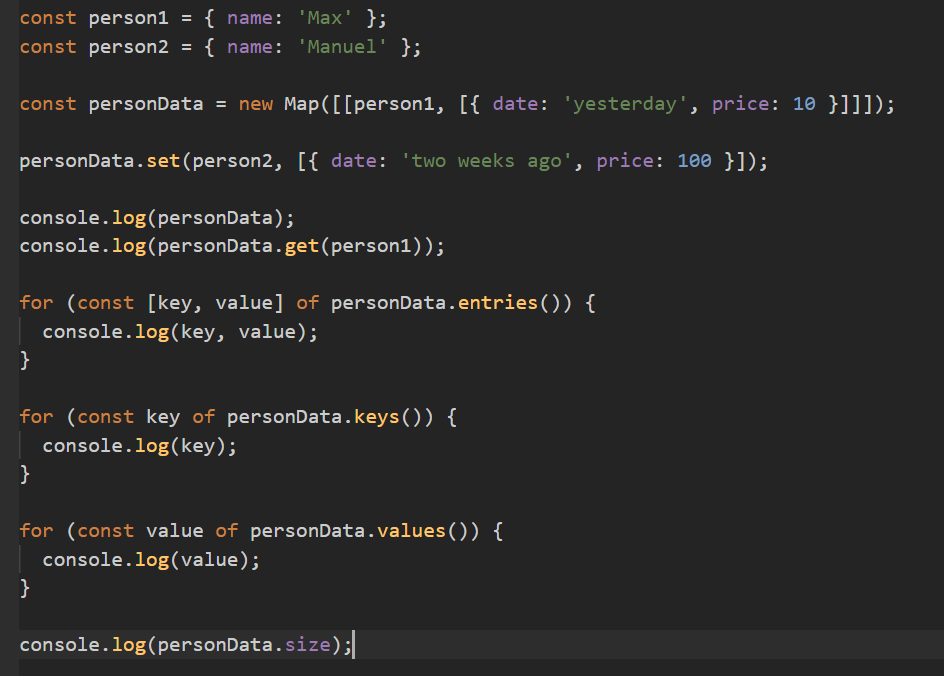
* 1. Sets and Maps



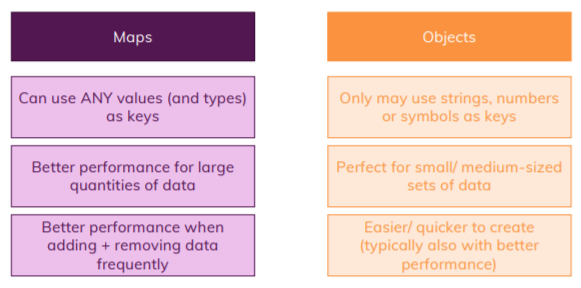
* 1. Sets
     1. Use to store unique data (not allow duplicate)



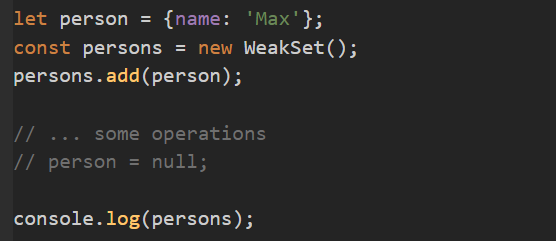
* 1. Maps

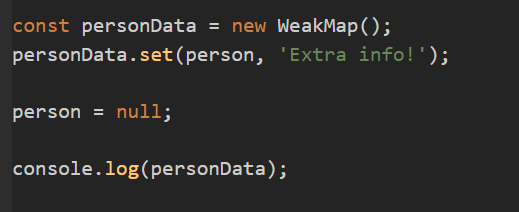


* 1. Maps Vs Objects

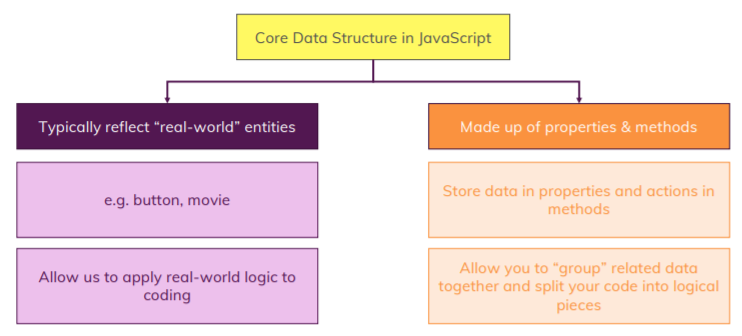


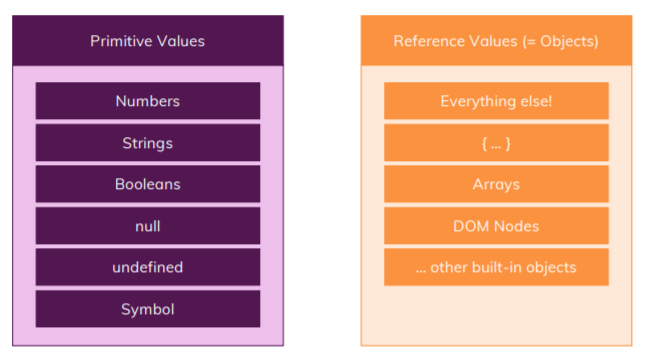
* 1. Understanding WeakSet and WeakMap
     + 1. weak set & map allows garbage collection to delete items that are part of the set as long as no other part of your code uses these items.

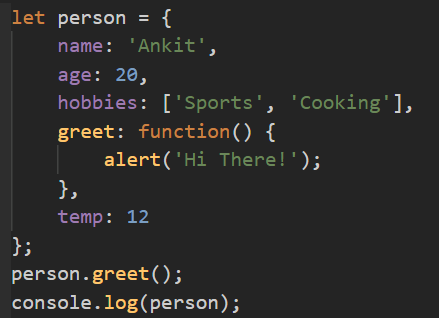




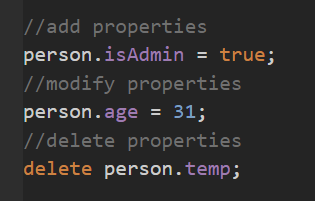
1. More on Objects
   1. What are objects



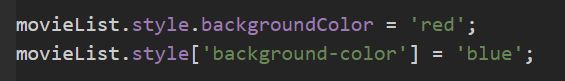




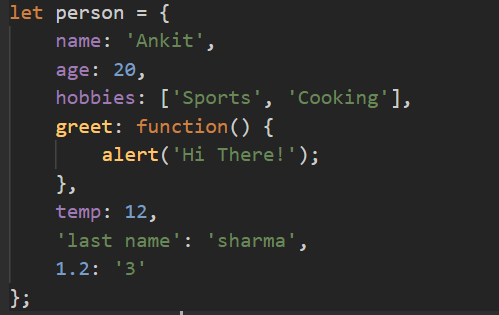
* 1. Adding, Modifying, Deleting properties

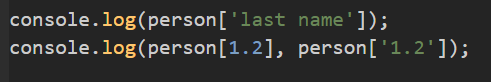


* 1. Special Key names & Square Bracket Property Access
     1. Keys by default converts into string. And we can pass key as string which can contains whitespace between then we have to access that properties with square bracket.

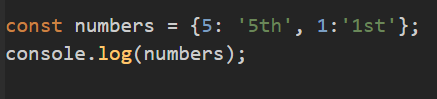


* + 1. Can pass number as key (Property)

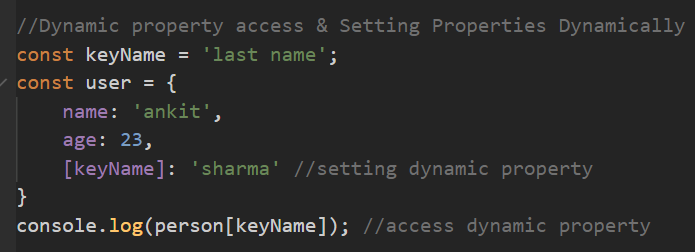




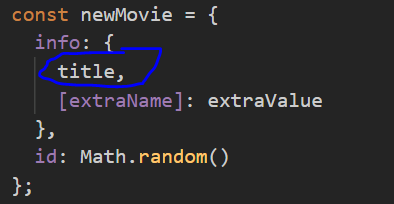
* 1. Property Types & Property Order
     1. If we pass property as both string and number then property will be in order as we added.
     2. If we pass property as only number then it will be sorted in ascending order (that why array index is sorted in ascending order as its an object)



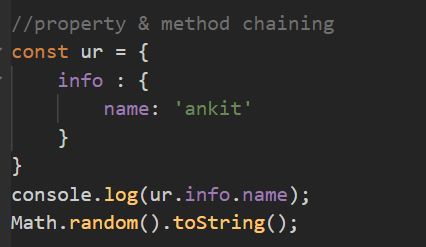
* 1. Dynamic property access & Setting Properties Dynamically



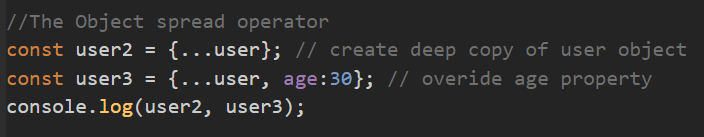
* 1. Property and value is same



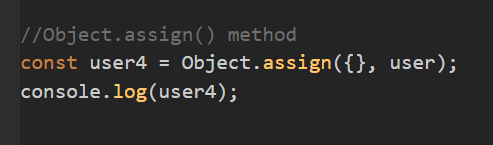
* 1. Property & Method chaining



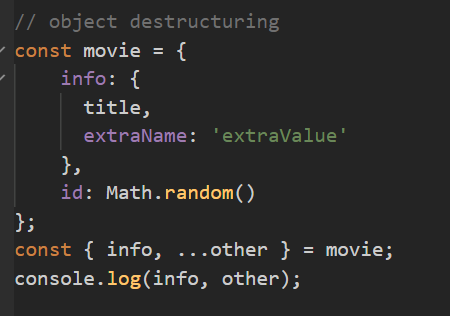
* 1. The Object Spread Operator (…)



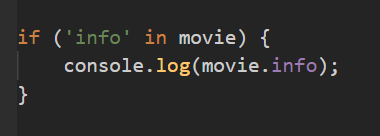
* 1. Understanding Object.assign() method



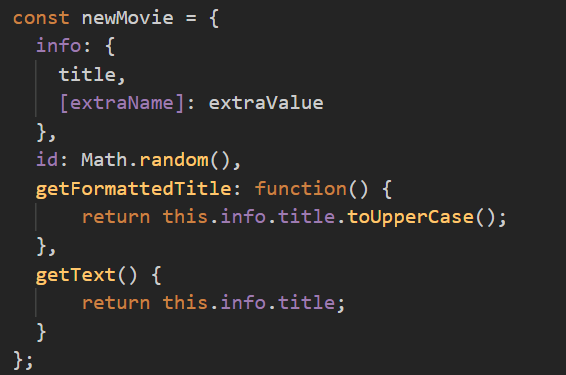
* 1. Object Destructuring
     1. Should use key as variable name



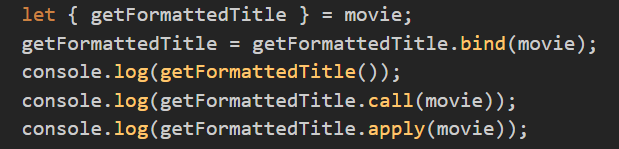
* 1. Checking for property existence



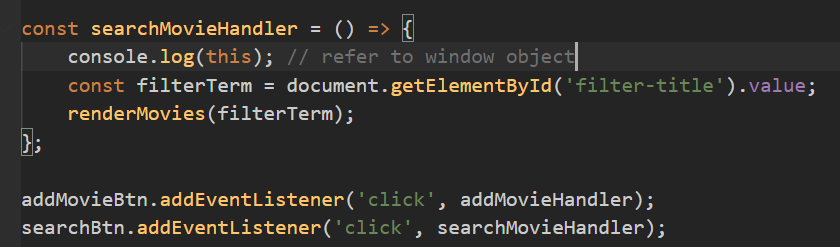
* 1. Introducing “this”
     1. Refers to current object



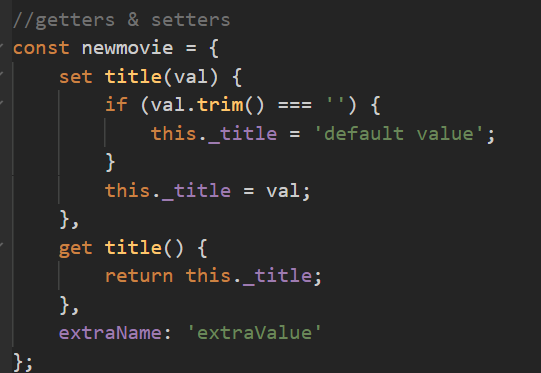
* 1. The bind(), call() and apply() method
     1. bind() is for binding this and call function later
     2. call() is bind this and call function at the same time
     3. apply() is same as call()



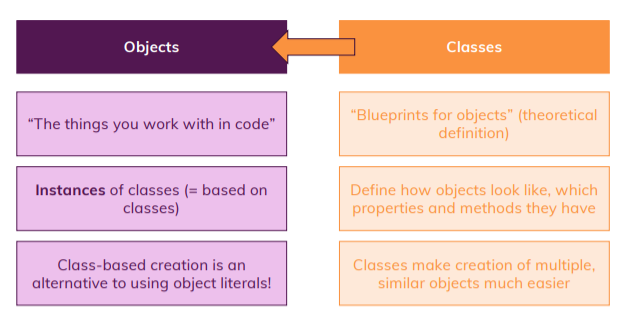
* + 1. The browser bind “this” for you (on event listeners) to the DOM element that triggered the event (button. Input etc).
    2. The “this” and arrow functions
       1. In arrow functions this refers to window object.

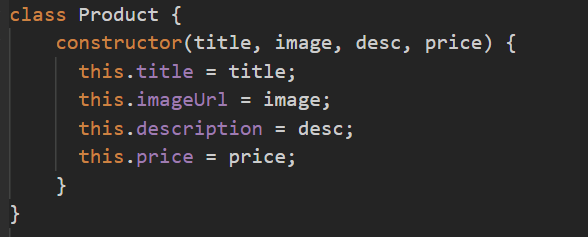


* 1. Getters & Setters



1. Classes & Oops Concepts
   1. Classes & Objects



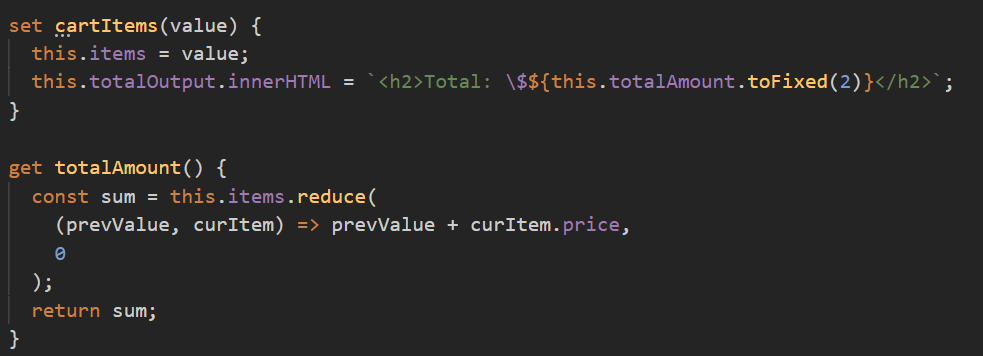


* 1. Static properties, fields and methods

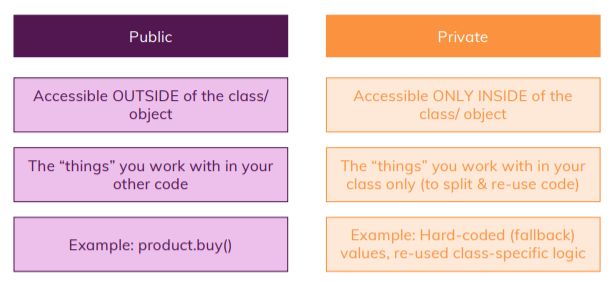




* 1. Getters & Setters

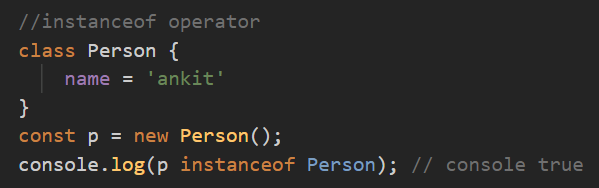


* 1. Inheritance
     1. Inherit properties and method using extends keyword
     2. super() use to call parent constructor
  2. Private properties, fields and methods
     1. Add # before field to make as private use same everywhere





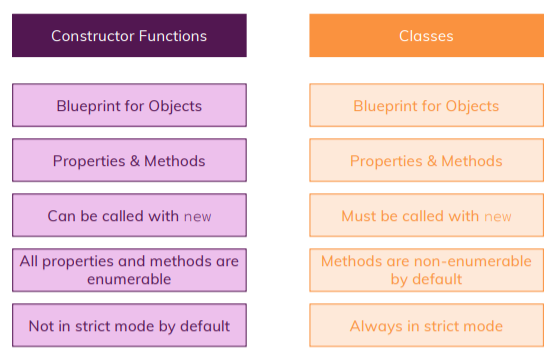
* 1. The instanceof operator
     1. Use to check object is instance of specified class or not



* 1. Object Descriptors
     1. It contains object metadata {configurable (for delete rights), enumerable (for-in loop rights), value, writable (for edit rights)}
     2. If we set value as false and then try to perform operation then it will not throw error.
     3. If we set enumerable as false then it will not print inside for-in loop.

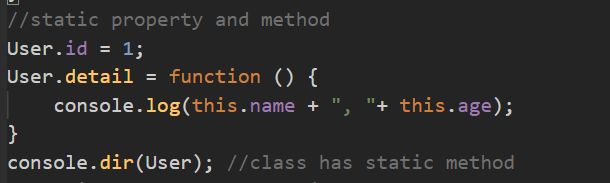


1. Prototypes & More
   1. Constructor Functions (function with new keyword)
      1. If we don’t use new while creating object then it throw error if we access properties.
      2. So we call function with new keyword then it provide feature or constructor function and return “this”.

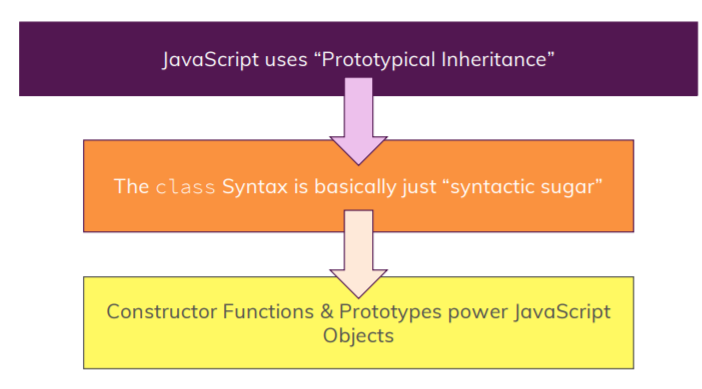


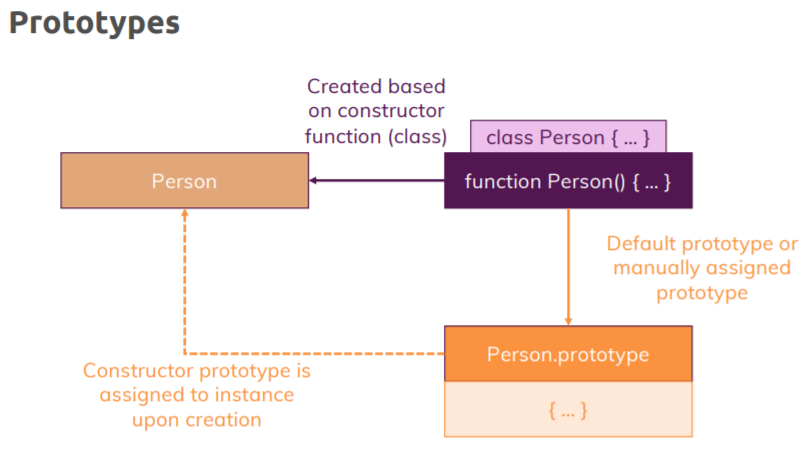


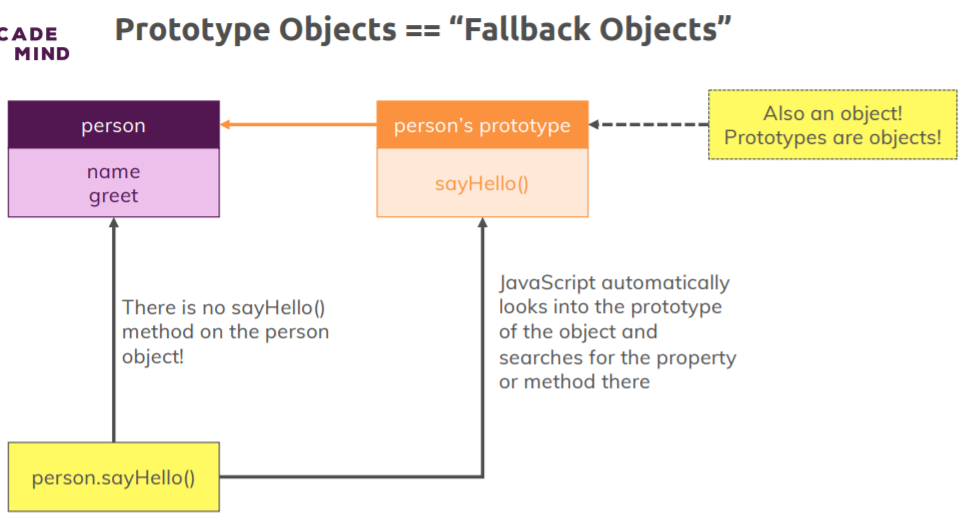
* + 1. The static property and methods in constructor function



* 1. Prototypes
     1. A prototype is an object that is linked to another object- it (the prototype object) kind of acts as a "fallback object" to which the other object can reach out if you try to work with a property or method that's not defined on the object itself.
     2. EVERY object in JavaScript by default has such a fallback object (i.e. a prototype object).
     3. Every object has \_\_proto\_\_ property
     4. Every Constructor function or Class has prototype property (ClassName.prototype) which is equal to (object**.\_\_proto\_\_**).
     5. The prototype property does something different: It sets the prototype object, which new objects created with this User constructor function will have.

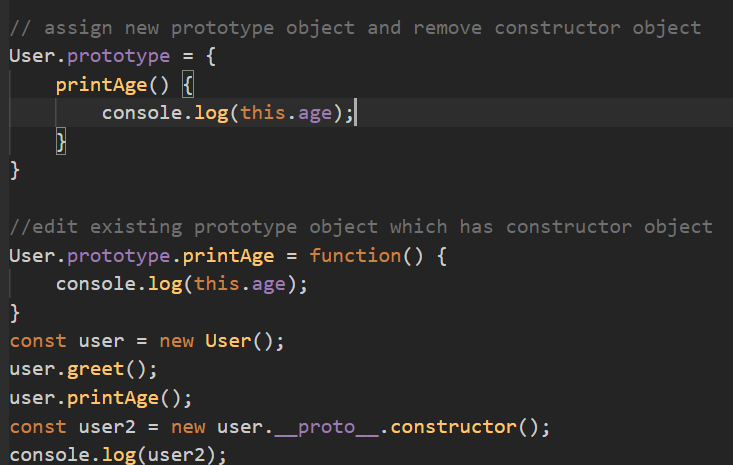




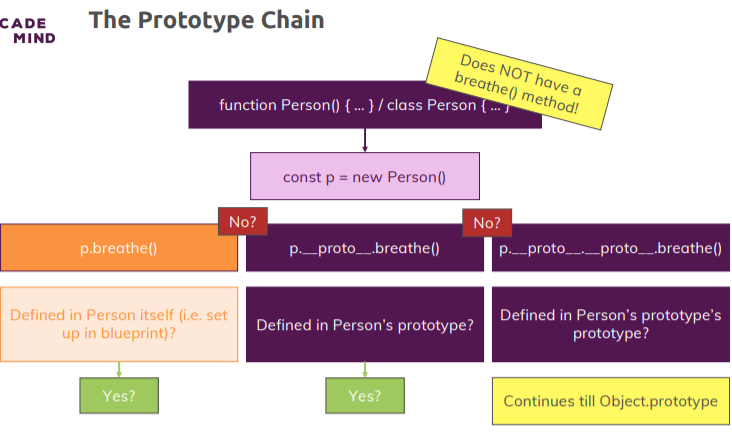




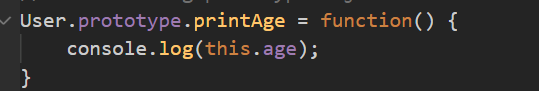
* 1. Editing prototype property
     1. If we add new prototype object then it will remove constructor object which added default.
     2. If we edit existing prototype object then it will keep constructor object as it is.

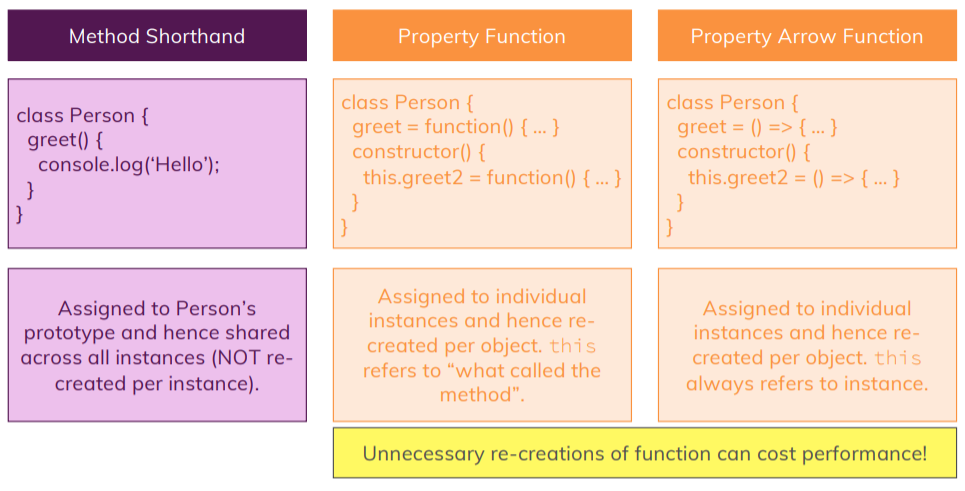


* 1. Prototype chain and The global object
     1. Object has prototype object which provide some static method.
     2. Class contain both static and non-static method and property where as object has non static method and property.

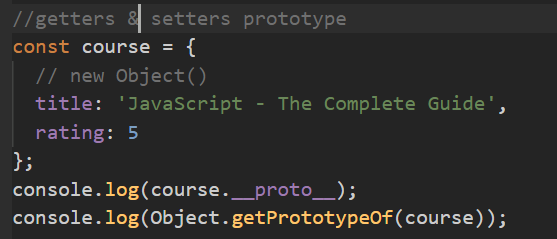


* 1. The prototype & method types
     1. In constructor function, to create function only once we can edit prototype and add new method into prototype directly which we reduce performance cost.

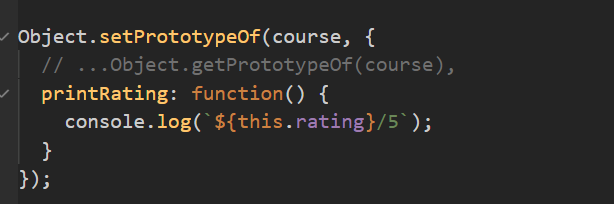




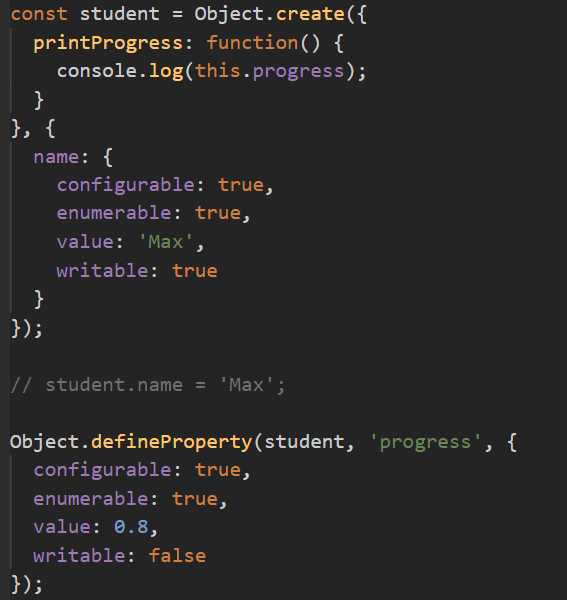
* 1. Array and String has their own prototype object.
  2. Getting & Setting prototypes after object has been created
     1. To getting prototype we can use two ways
        1. ObjectName.\_\_proto\_\_
        2. Object.getPrototypeOf(ObjectName)



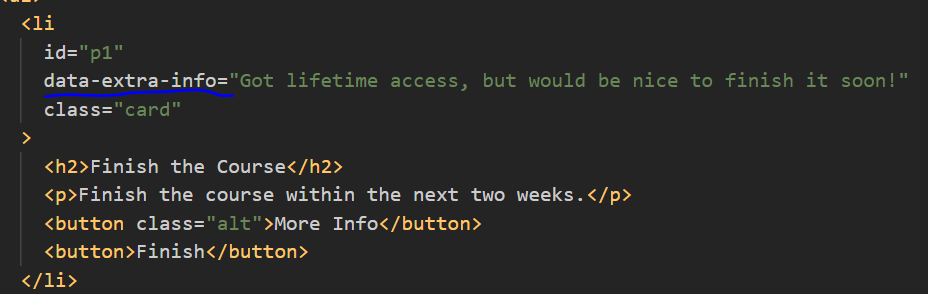
* + 1. To setting prototype
       1. We can use Object.setPrototypeOf(objectName, {}) method

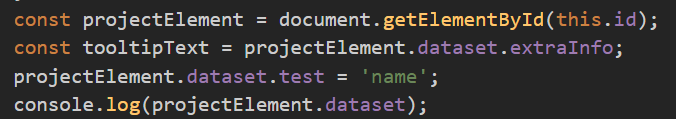


* + - 1. We can use Object.create() method

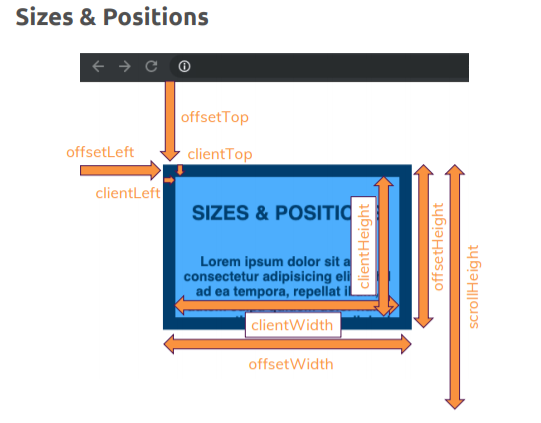


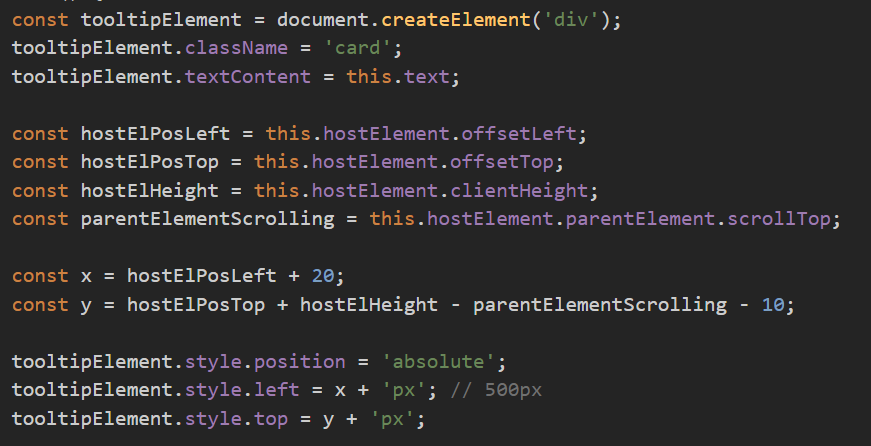
1. Advance DOM Concepts
   1. Using dataset (data-\* attribute)
      1. Use to set some data in html tags. (document.getElementById("id").dataset)
      2. We can set data in html tag directly or we can set using java script as well.





* 1. Getting element box dimensions
     1. In browser console, using $0 we can access current selected DOM element.
     2. And same properties we can use in an DOM element.
     3. There are lot of attributes like:
        1. $0.dataset : provide dataset (data-\* attribute)
        2. $0.getBoundingClientRect() : provide box dimensions
        3. Access other with below property

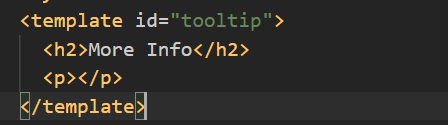




* 1. Scroll handling
     1. $0.scrollTo(0, 50) : scroll y to 50 px (scroll from top position)
     2. $0.scrollBy(0, 50) : scroll y to extra 50px (scroll from current position)
     3. $0.scrollIntoView(): scroll to show new changes. Also pass behavior for animation.

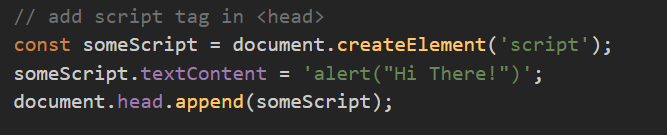


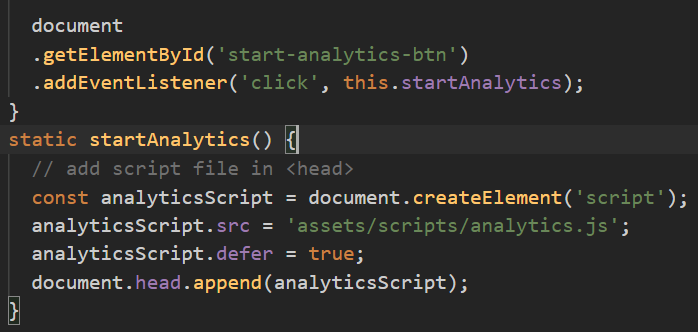
* 1. Working with <template> tags
     1. <template> tag won’t render in browser on load.
     2. It can be use to write some html code and later use in some other place like tooltip or modal. It will allow us to keep all html code in html file instead of java script code.



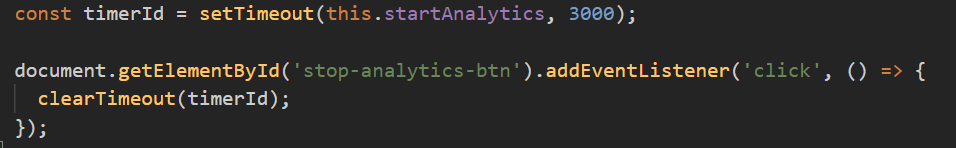


* 1. Loading script dynamically

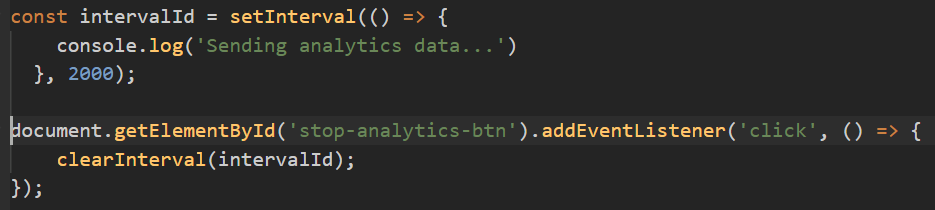




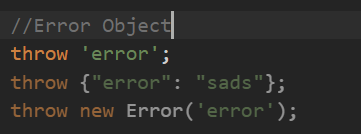
* 1. Setting Timers & Intervals
     1. Timers run after some time only once



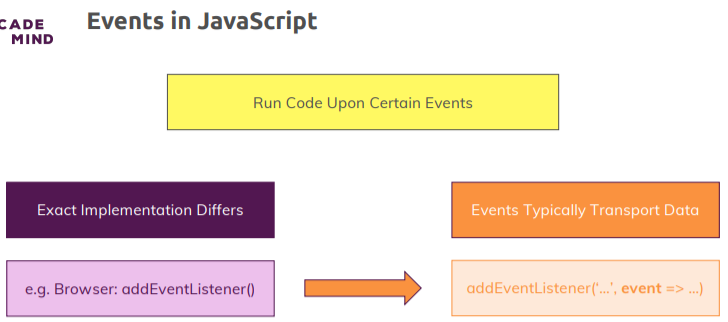
* + 1. Intervals run after some specified time until with clear intervals.



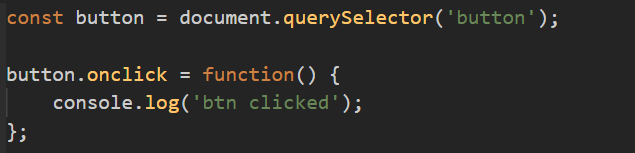
* 1. The “location” & “history” object
     1. The “location” object can be use to navigate to different page
     2. It provide information current active tab.
  2. The “navigator” object
     1. It provides information about the browser and system.
     2. Browser detect and return a string (<https://developer.mozilla.org/en-US/docs/Web/API/Window/navigator#example_1_browser_detect_and_return_a_string>)
     3. Browser detection using the user agent (<https://developer.mozilla.org/en-US/docs/Web/HTTP/Browser_detection_using_the_user_agent>)
  3. Working with Date object
     1. It provides information about date and time
     2. <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date>
  4. The “Error” Object & Constructor function

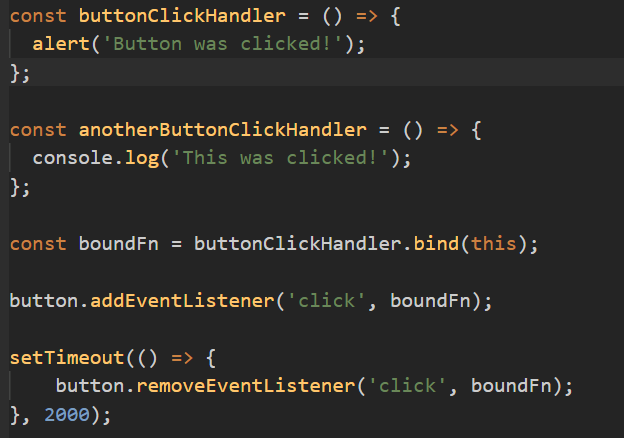


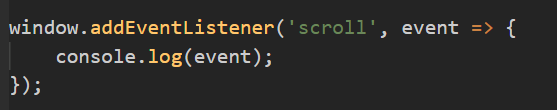
1. Working with events



* 1. Add and remove event listeners
     1. To remove event listener we need to store function in constant.



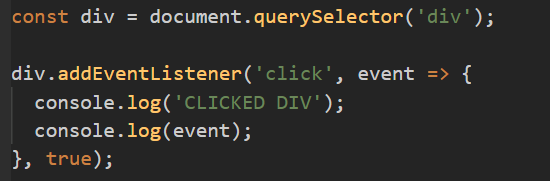




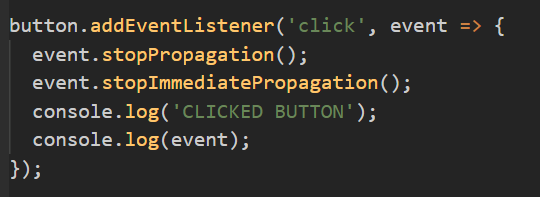
* 1. The preventDefault() method



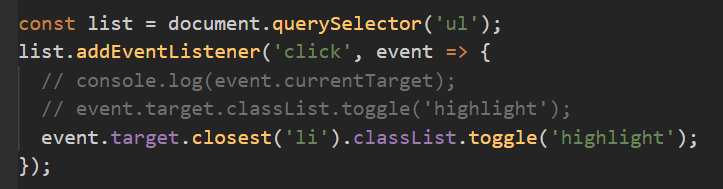
* 1. Understanding “Capturing” & “Bubbling” phases
     1. The browser runs through two phases where it checks for listeners to that event (Capturing and Bubbling Phases).
     2. The capturing phase goes from outside to inside and the bubbling phase on the other hand does the opposite, it goes from inside to outside.
     3. All event listeners you add with add event listener are by default registered in that bubbling phase.
     4. We can register event for capturing phase by passing true in addEventListener()



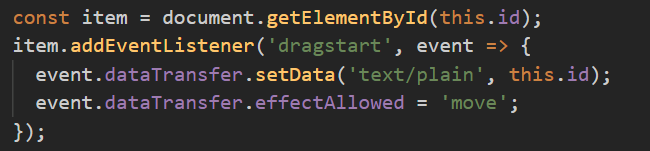
* + 1. Using **stopPropagation**() we can stop propagating of event listener from inside to outside.
    2. If we have multiple event listener for same element, in that case also we can stop propagation using **stopImmediatePropagation**() method.

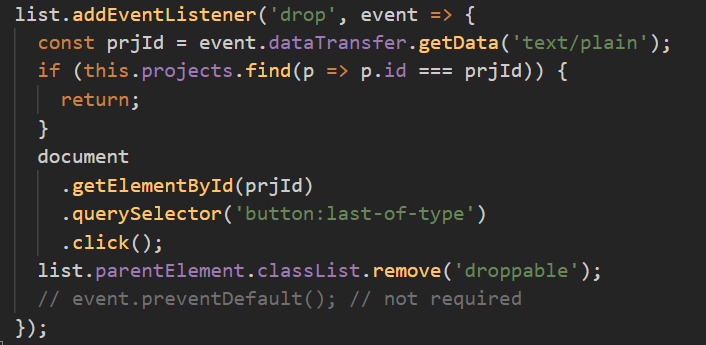


* + 1. To check event will propagate or not (event.bubbles). some event won’t propagate like mouse enter, drag and drop etc.
  1. Using Event Delegation
     1. With event propagation, you can do quite interesting things, specifically you can implement a pattern which is also called event delegation.

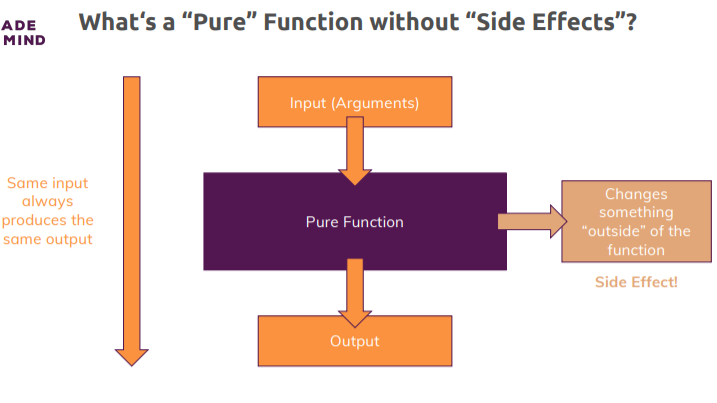


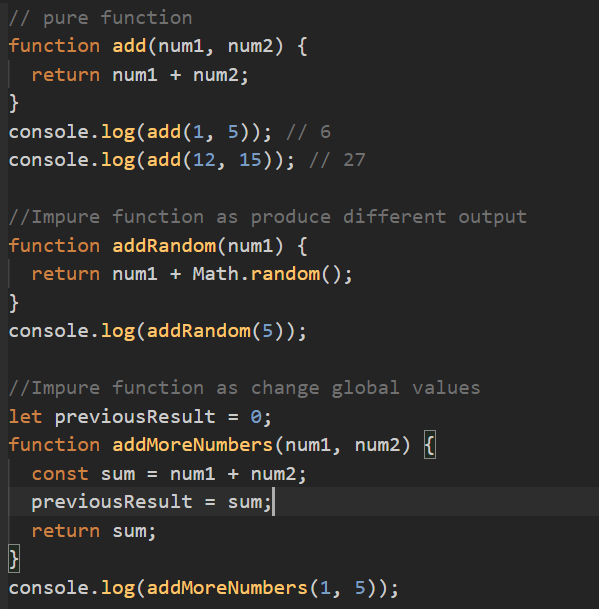
* 1. Event handler function & this
     1. Inside arrow function, this refers to window object
     2. Inside normal function, this refers to current target
  2. Drag & Drop
     1. Mark elements as ‘**draggable’**, which will be drag
     2. Listen to “**dragstart**” event (describe operation & append data)
     3. Accept drop via “**dragenter**” and “**dragover**” events => preventDefault()
     4. **Optional**: listen to “**dragleave**” event
     5. Listen to “drop” event & update data/UI
     6. **Optional**: listen to “**dragend**” event & update data/UI
     7. Use setData to pass some data and which will extract in drop point



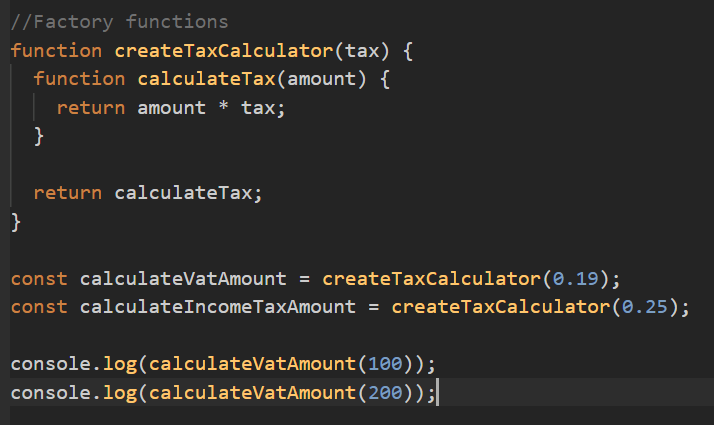


1. Advance Function Concepts
   1. Pure Functions & Side Effects
      1. The function which produce same output for same input and don’t change something outside of the function(which is called side effects).

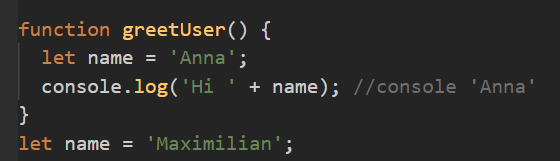




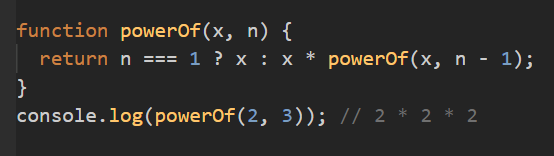
* + 1. Factory Functions
       1. Factory functions is that you have a function that produces another function.



* + 1. Closures
       1. Every function in java script is closure.
       2. Because every function closes over the surrounding environment which means it registers the surrounding environment and the variables registered there and it memorizes the values of these variables.
       3. Inner variable won over outer variable.



* + 1. Recursion
       1. Functions call itself is called recursion



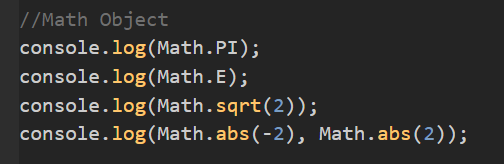
1. More on Numbers & Strings
   1. Numbers
      1. Every numbers in java script is a float e.g 5, -3, 12 is 5.0, -3.0, 12.0
      2. Numbers are stored as 64 bit floating point in java script
      3. Number.MAX\_SAFE\_INTEGER & Number.MIN\_SAFE\_INTEGER (maximum and minimum possible integer value in number)
      4. Number.MAX\_VALUE & Number.MIN\_VALUE (maximum and minimum possible

value in number)

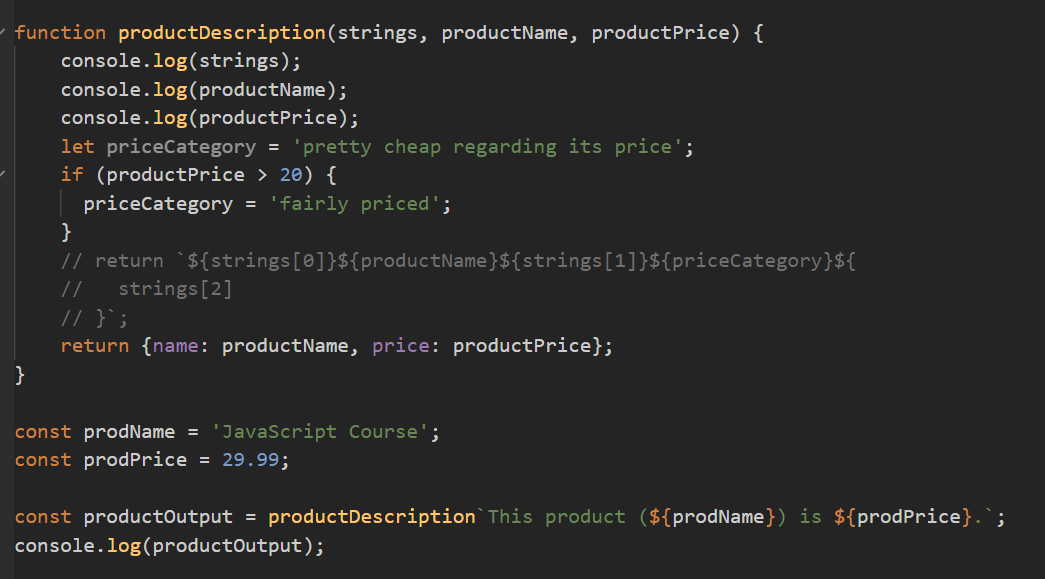
* + 1. Number.toString(radix): using to we can convert decimal into binary or any other by pass radix. E.g. (4).toString(2) = “100”
    2. Number.toFixed(fractionDigits): use to set no of digit after decimal point
  1. BigInt Type
     1. If we want to store value more than Number.MAX\_SAFE\_INTEGER or less than Number.MIN\_SAFE\_INTEGER then we need to use BigInt (by adding n in the last of number e.g. 900719925474099676n, -900719925474099676n)
     2. It only accept integer, no decimal places.
  2. The global Number and Math object
     1. Number.isFinite(no): use to check number is finite or infinite.



* + 1. Using Math object we can get some predefined value like Math.PI, Math.E etc.



* 1. Tagged Template
     1. A tagged template is in the end a function that works together with a template literal. We add a template literal right after our function name (fun\_name ``).



* 1. Regular Expressions (i.e. RegEx)
     1. Used to search some pattern in string.

