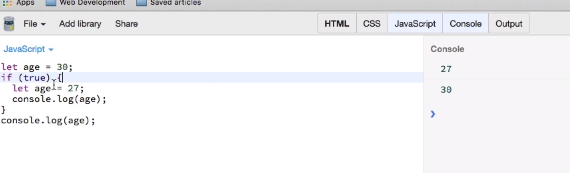
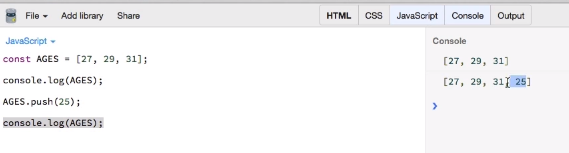
**JavaScript ES6**

Section 1: By Maxmillan

1. Syntax changes and Extensions; New Keywords
2. let – let allows us to use block scoping, which means any variable defined inside the for or if will be available only inside the scope of that for or if. However, with var its not possible. In case of var only function scope will work.



1. const: const has also block scope. If we put an array in the const then we push any item in that array, it will get created without giving any error:



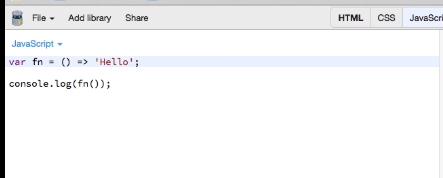
A const or var or let does not contain array they contain reference to the array. We are just changing the value where the pointer points.

If there is a cont object then also we would be able to change the object.

1. Hoisting with let and const:

Variable hoisting will not work with let and const.

1. Fat Arrow Functions

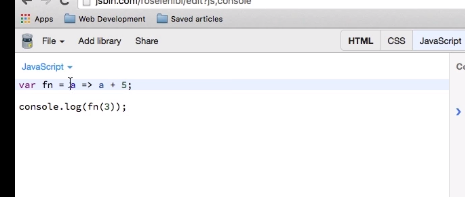


Above is the scenario when the function will only return one word, then we don’t need to write return also.

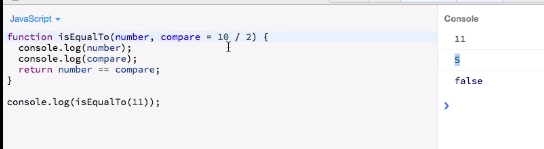


2 Functions in above screenshot are same to each other.

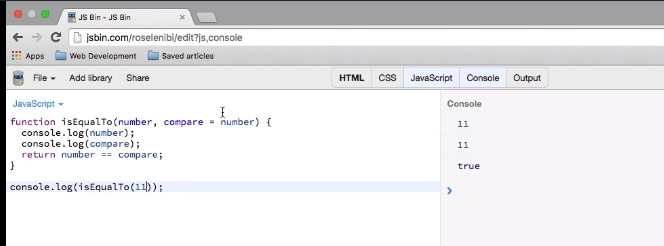
1. If you have 1 parameter passed in function, then we can leave out the parenthesis.

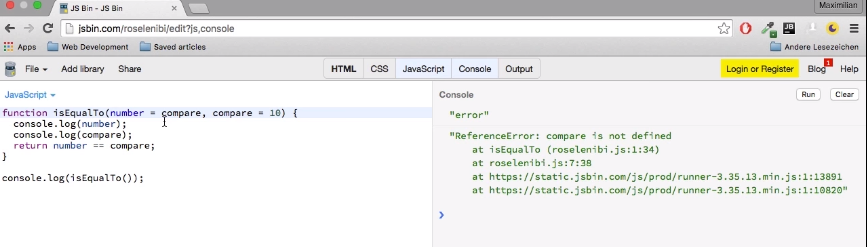


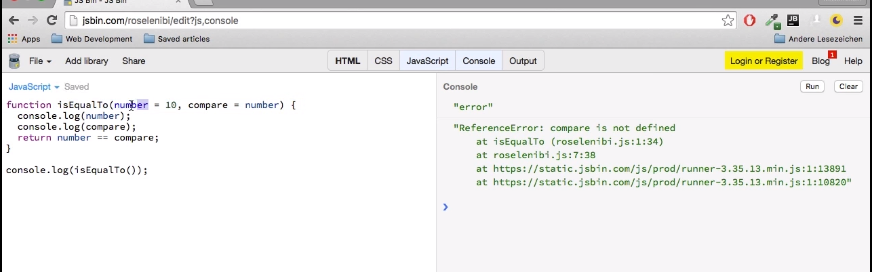
1. Fat Arrow Functions and “this” – For arrow functions its always Global object i.e. this function.
2. Default Parameters -> we can provide default value to the parameters of a function in ES6.
3. 
4. Below expression will also run fine:



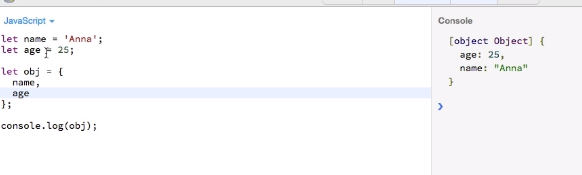
1. Below function arguments are also valid:



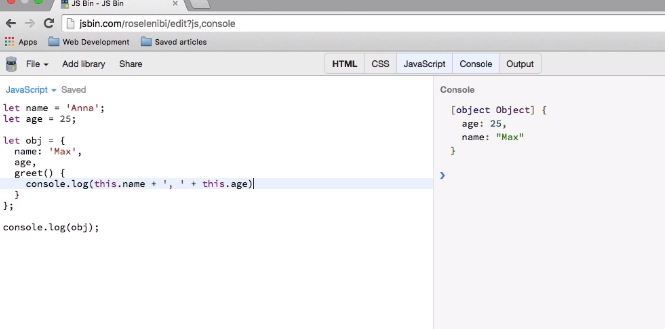
1. Here first assignment of the argument is not defined.
2. 
3. Below one will work:



1. Object Literal Extensions:
2. ES6 object declaration when Property and Value of the object are same for an item:

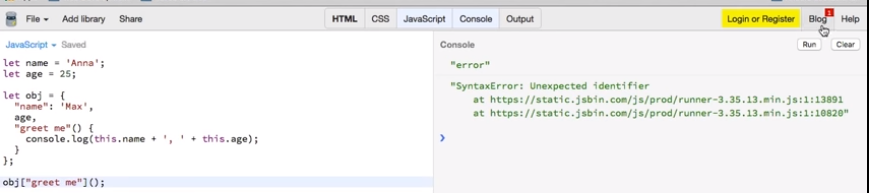


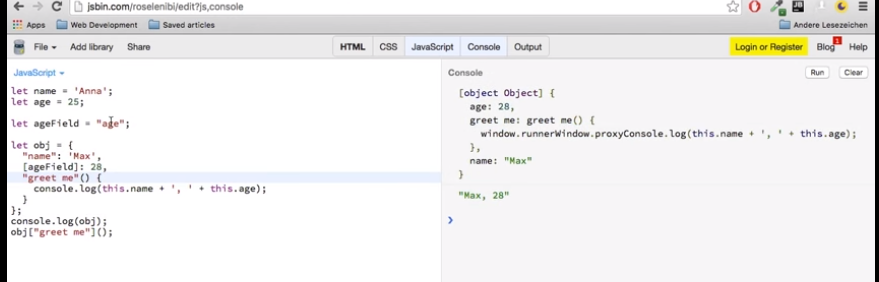
1. 
2. New syntax for the property function of the object:



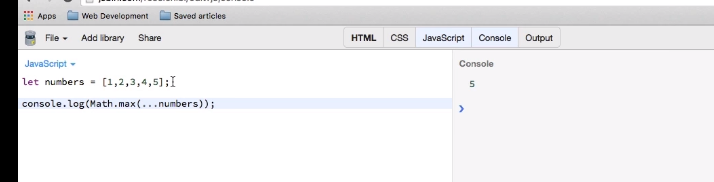
1. We can write property name in the object as string also:



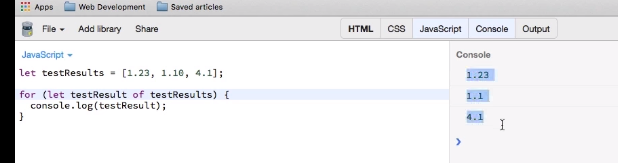
1. 
2. Dynamically add value to the object:



1. The REST operator  
   
2. The Spread operator:

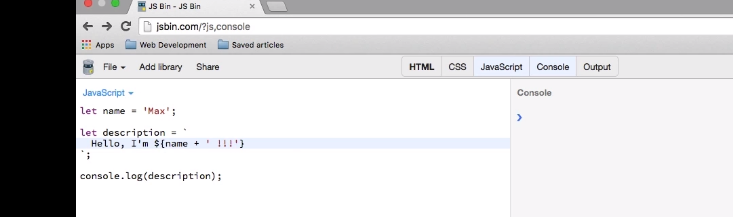
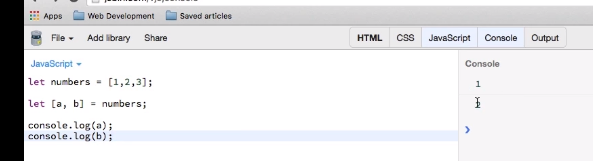


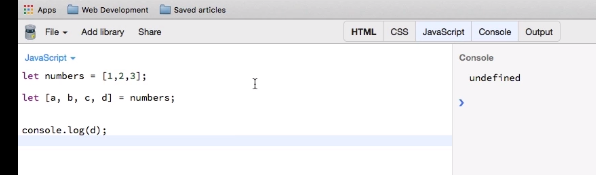
1. The For-of-Loop



1. Template literals – Strings with extra features



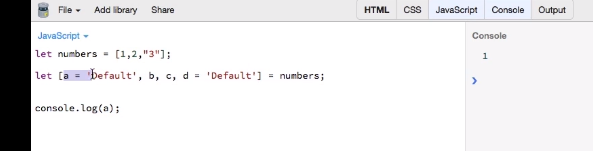
1. 
2. De-structuring - Arrays
3. 
4. If we try to fetch more values than the array has:



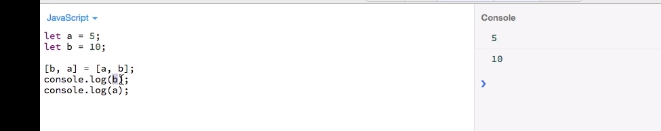
1. Pulling out remaining values using rest operator as a parameter:



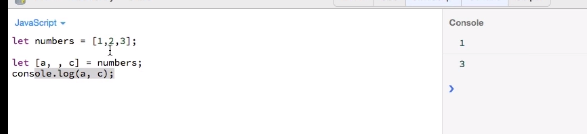
1. Mixture of default values and De-Structuring here:



1. De-structure to Swap variables:

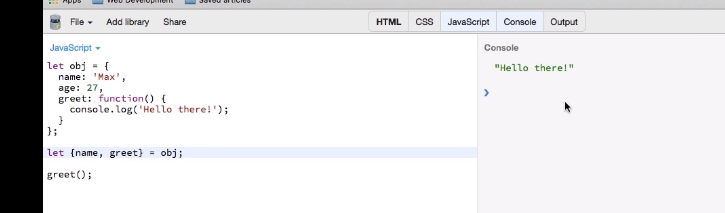
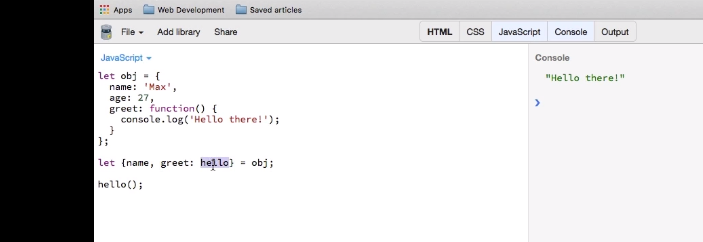


1. Ignore certain values:



1. Immediate de-structure after creation:



1. de-structuring the objects:
2. 
3. 
4. Using alias for the object property names:
5. 

Section 2: