1. Light weight, OO programming language.
2. Datatypes are numbers, strings, Boolean, undefined (value not assigned to variable yet) and null (non-existent). JS is dynamically typed. Camel case is used for naming variables. JS automatically does ***type coercion*** i.e.  
     
   **This gives an error in Python**

age = 28

name = 'Jaydeep'

print(name + ' ' + age)  
  
**BUT WORKS PERFECTLY FINE IN JS BECASUSE OF TYPE COERCION**age = 28

name = 'Jaydeep'

console.log(name + ' ' + age);

***== will perform type coercion eg. 23 == ‘23’ will be true.***

***=== will not perform type coercion eg 23=== ‘23’ will be false because string not equal to integer***

**IF-ELSE**

Takes the following forms  
  
if(condition){

}

If(condition){  
}else{

}

if(condition){

}else if(condition){

}else{

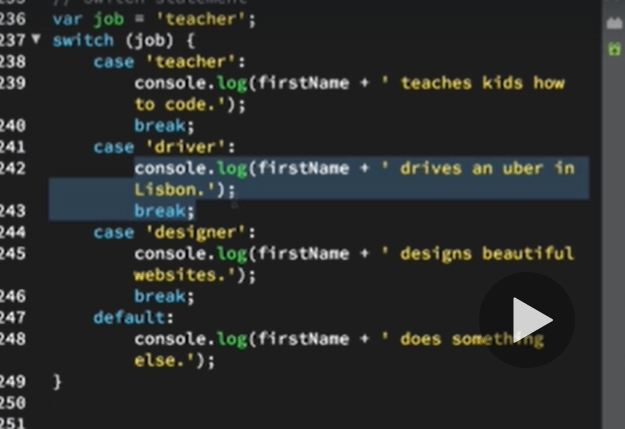
}

**TERNARY OPERATOR**

1. This is still like an if-else block but all in one line and take the following form  
     
   if condition ? do something : else do something else

Age >= 18 ? console.log(can drink) : console.log(cannot drink)

**SWITCH**



**TRUTHY AND FALSEY**

1. Falsey values are : NULL, UNDEFINED, NaN, 0, ‘’
2. TRUTHY values are all those which are not FALSEY.

**FUNCTIONS**

1. Use camelCase for naming.

**Arrays**

1. var nameArray = [‘Tinkya’,’Jaydeep’,’Chaitu’];
2. var nameArray = new Array(‘Tinkya’,’Jaydeep’,’Chaitu’);
3. zero indexed.
4. Arrayname.length will give length
5. **Push()** can be used to add element to the end of the array.
6. **Pop()** will remove the last element.
7. **Unshift()** will add element to start of array
8. **Shift()**  will remove the first element
9. **indexOf(element)** will give index of element which is passed and returns -1 if element is not present. A nice use case of this is as below  
     
   var isDesigner = johnArray.indexOf(‘designed’) === -1 ? ‘John is a designer’ : ‘John is not a designed’  
     
   console.log(isDesigner)

**Objects**

1. Javascript objects store data in key value pairs.  
     
   var jaydObject = {  
    name : ‘Jaydeep’,  
    age : 30,  
    profession : ‘’DESIGNER’,  
    family : [‘Jaywant’,’Chaitu’,’Mummy’,’Tinkya’]  
   }
2. to change values we can simply reference the object as jaydObject.profession = ‘FOOTBALLER’ or jaydObject[‘age’] = 900
3. Another way to create a object is   
   var jaydObject = new Object()  
   jaydObject.name = ‘Jaydeep’  
   jaydObject.profession = ‘DESIGNER’
4. var jaydObject = {  
    name : ‘Jaydeep’,  
    birthYear : 1989,  
    profession : ‘’DESIGNER’,  
    family : [‘Jaywant’,’Chaitu’,’Mummy’,’Tinkya’]  
    calcAge = function(){  
    return 2019 – this.birthYear;  
      
   }  
   }

console.log(jaydObject.calcAge()); will then give age as this.birthYear will use the birth year from jaydObject

1. var jaydObject = {  
    name : ‘Jaydeep’,  
    birthYear : 1989,  
    profession : ‘’DESIGNER’,  
    family : [‘Jaywant’,’Chaitu’,’Mummy’,’Tinkya’]  
    calcAge = function(){  
    ***this.age = 2019 – this.birthYear;***   
   }  
   }

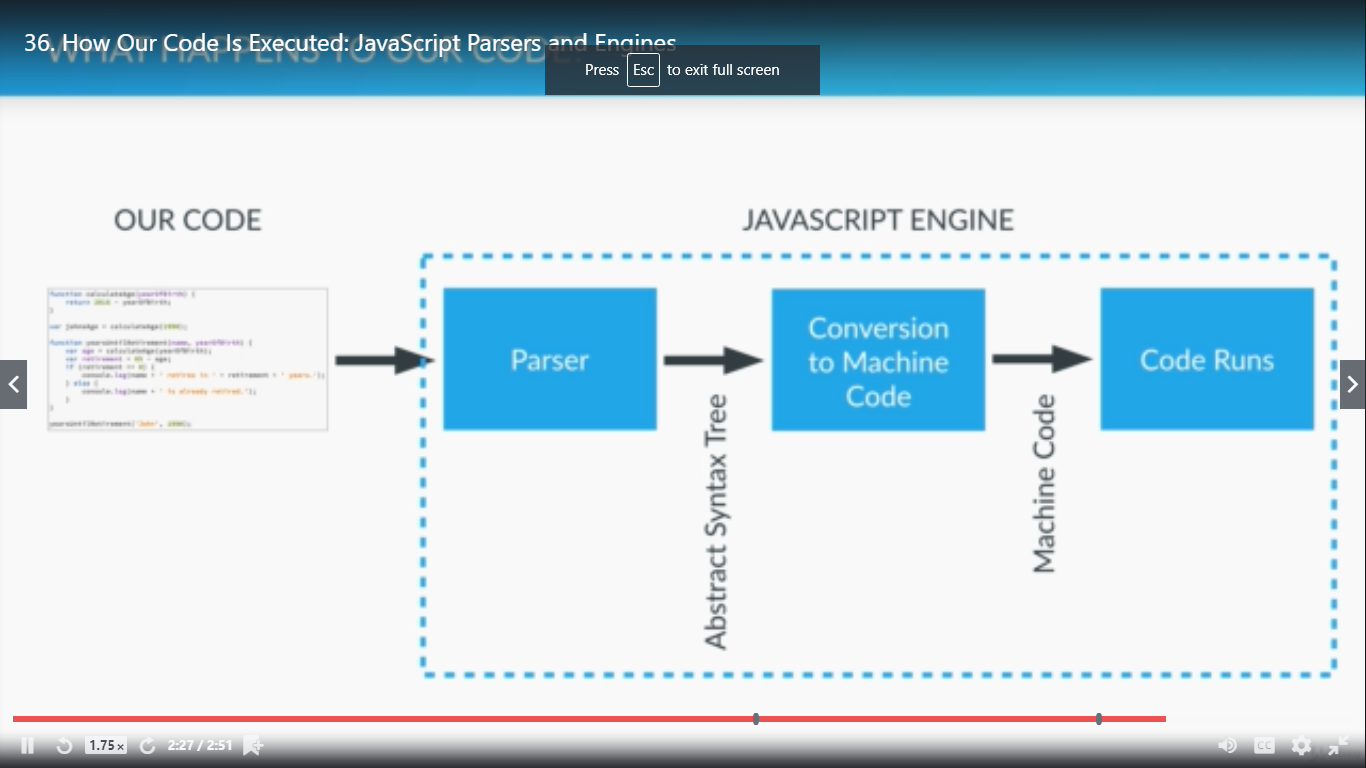
jaydObject.calcAge() – notice the use of this keyword and now when we output the same age will be a field in the jaydObject

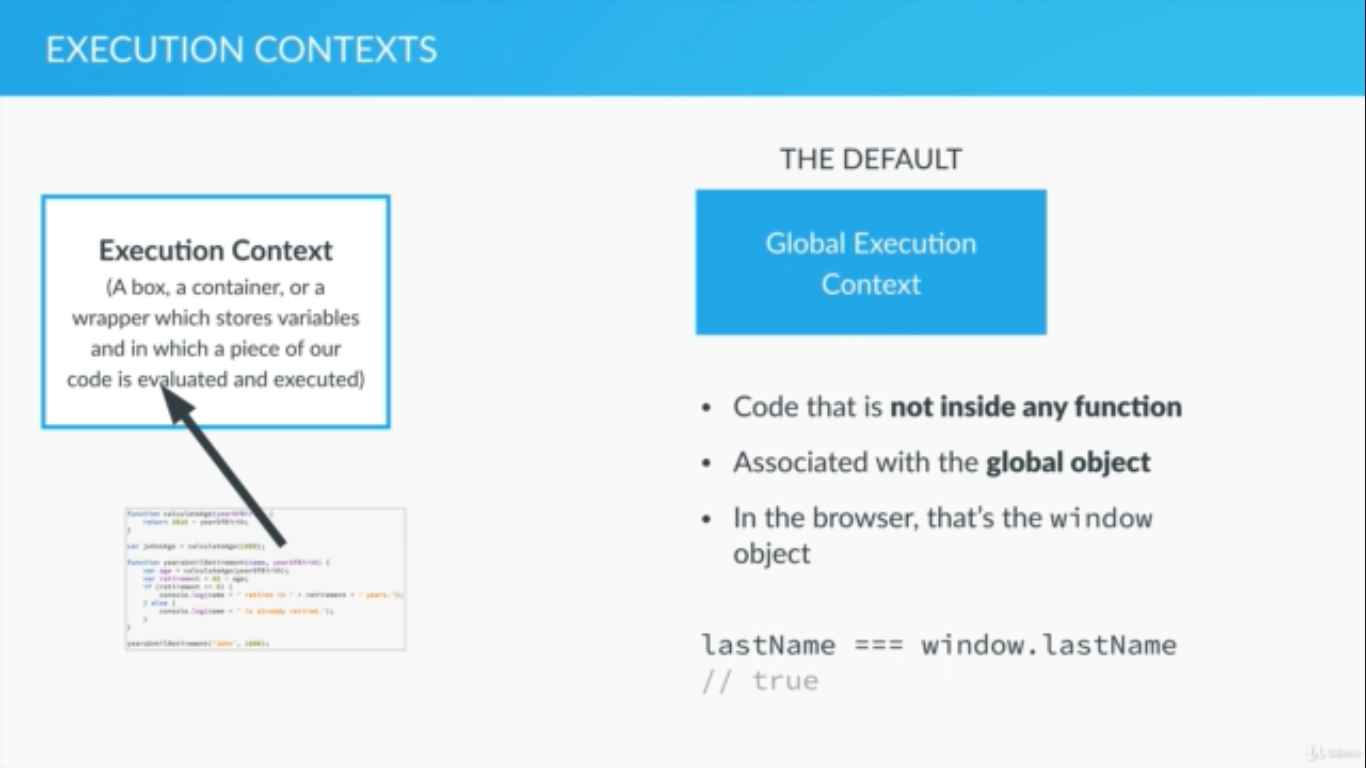
*console.log(jaydObject)  
  
{  
 name : ‘Jaydeep’,  
 birthYear : 1989,  
 profession : ‘’DESIGNER’,  
 family : [‘Jaywant’,’Chaitu’,’Mummy’,’Tinkya’]  
 age : 31* *}  
}*

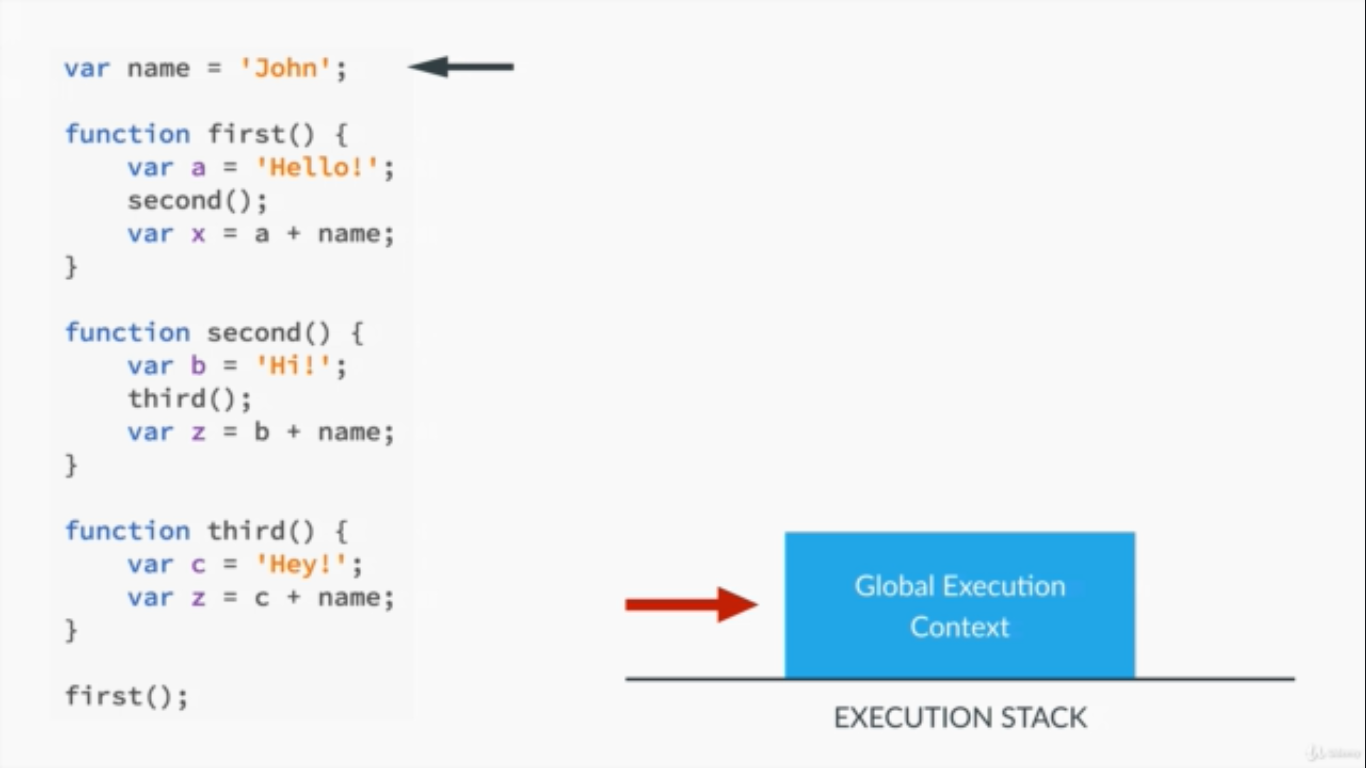
**LOOPS**

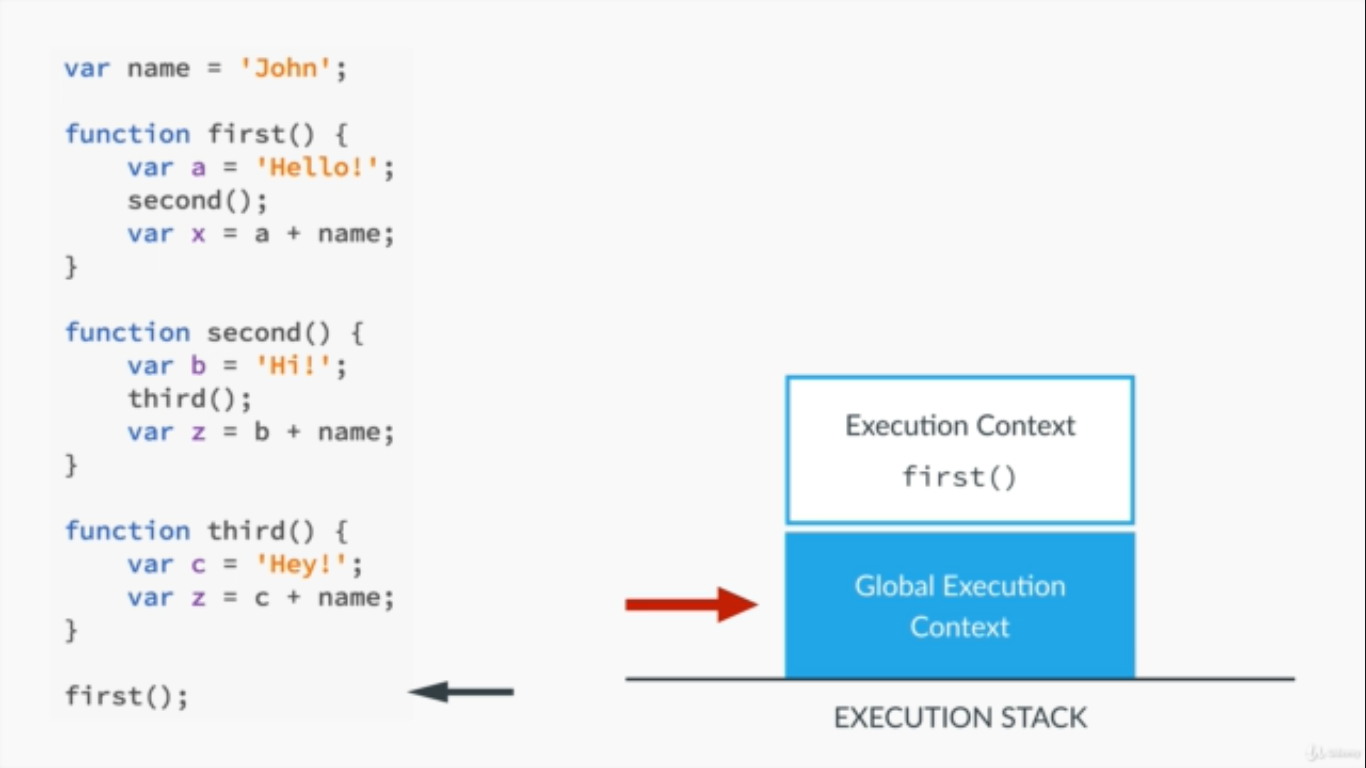
1. ***for (var I = 0; i< 10; i++){}***
2. ***typeof*** keyword is used to determine the type of the data.

**BTS OF HOW JS IS EXECUTED**

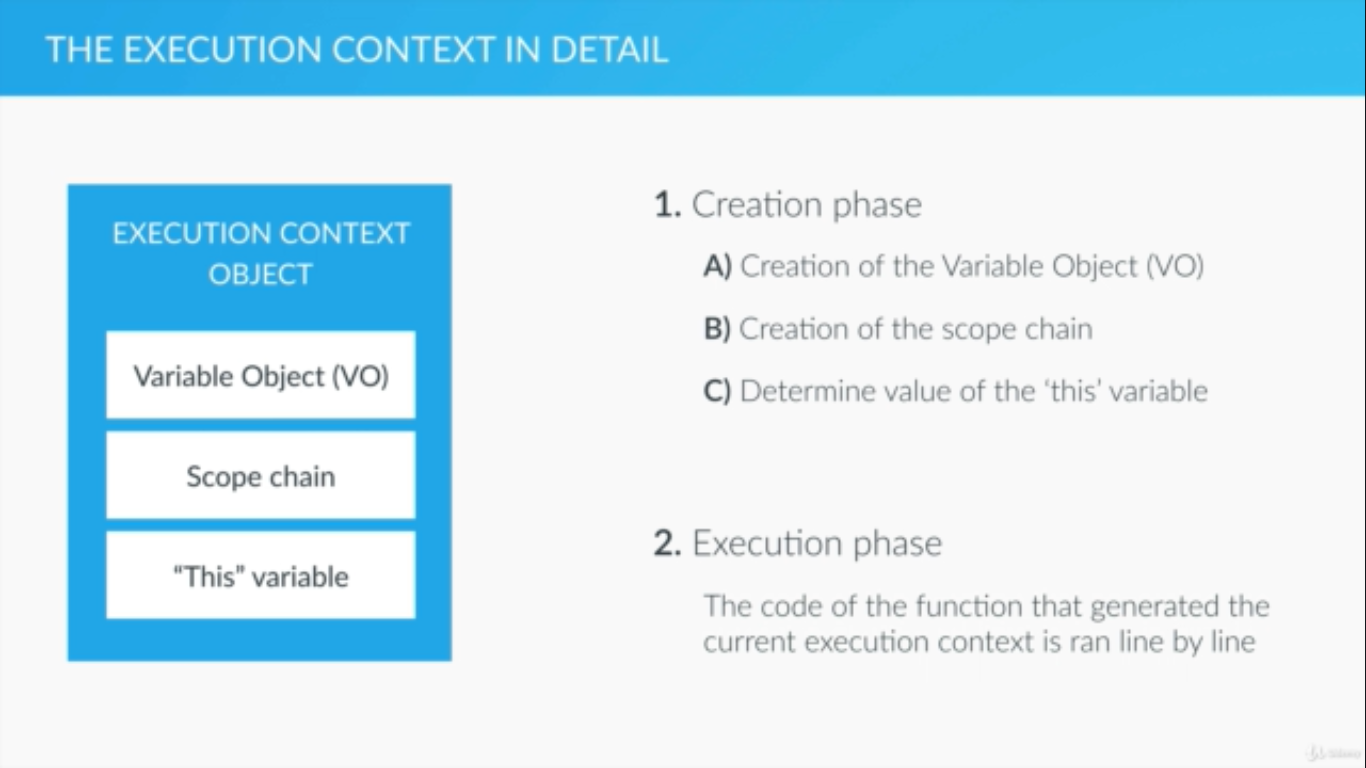


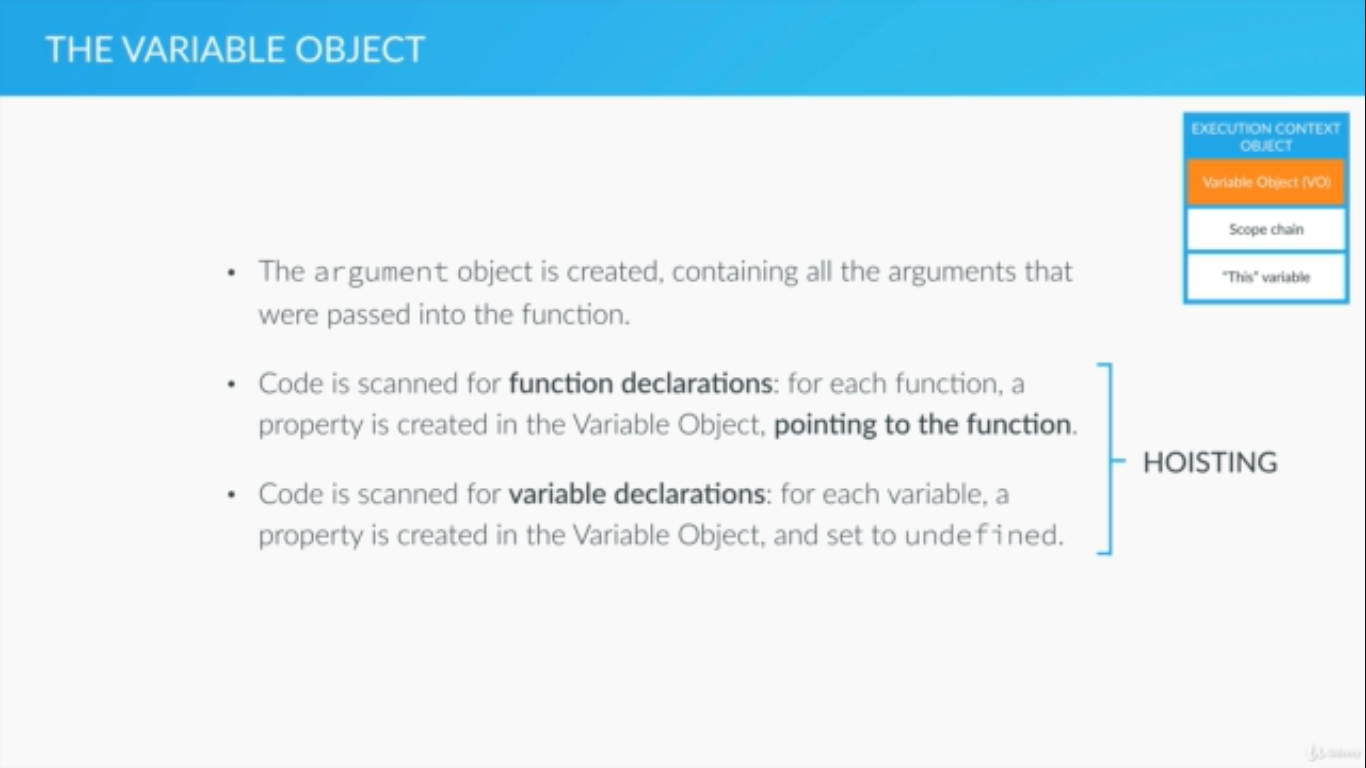


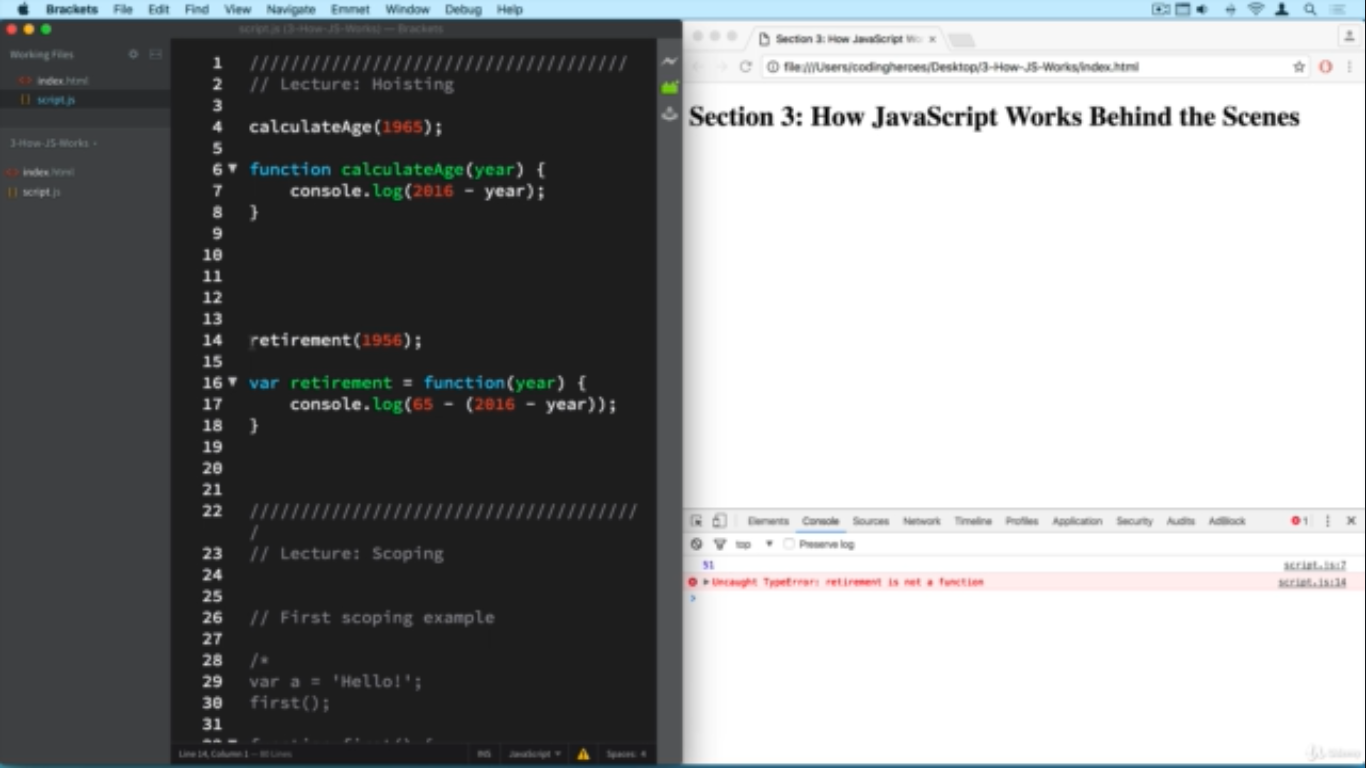


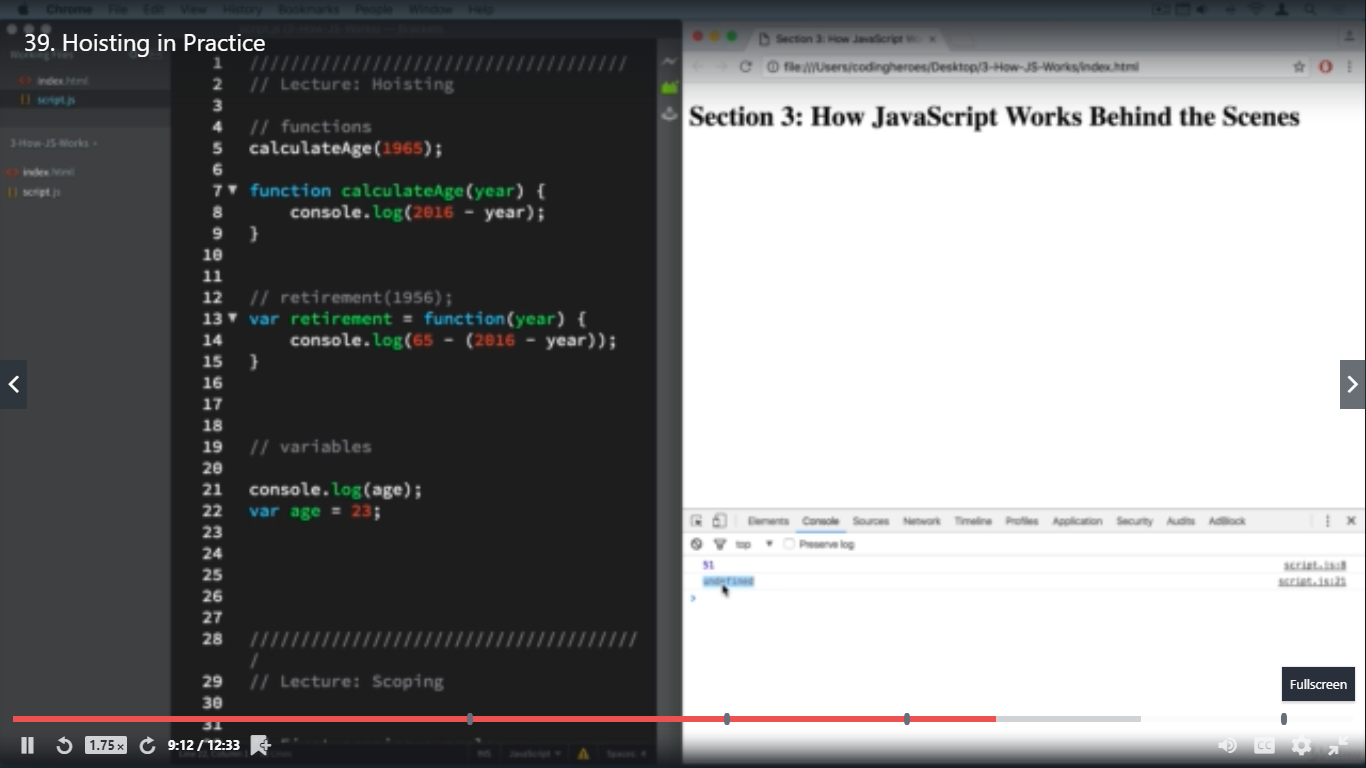












**In the above screenshot the result is UNDEFINED because JS know that variable age exists bt just doesn’t know it’s value yet.**

