**WORKING OF JAVASCRIPT:**

Execution of javascript happens in Execution context (which stores variables and functions).  
If there is no separate function, Global execution context is used, aka window object.

Whenever a function is called, a new execution context is created and is added on top of the execution stack.

Each execution context has two phases in which it works :

1. Creation phase: It in turn has three phases:
   1. Variable Object
      1. Arguments passed to the function is saved into newly created argument object.
      2. Function declarations are scanned and a property Is created for each pointing to the respective function.
      3. Variable declarations are scanned and then each newly created property is set to undefined.

The reason why calling a function before its declaration works fine in JS is that the creation phase is performed first, which declares and defines all the functions first and only after this, execution phase is started where the commands are executed line by line.

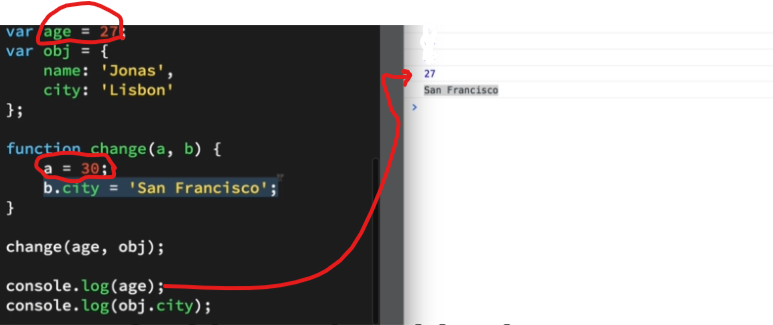
However this wouldn’t work with function expressions like   
var retirement = function(years){return 65-years; }  
since this is not a function declaration, it is not stored in variable object during creation phase.

When a variable is accessed before its declaration, it returns undefined and if it is accessed without declaring it anywhere in the file, it returns error saying it is not defined.

* 1. Scope Chain: scope defines an environment where a variable can be accessed.  
     works in a hierarchical way, hence a function inside a function can access parent function’s variables, but not the other way round.
  2. “This” Variable – it

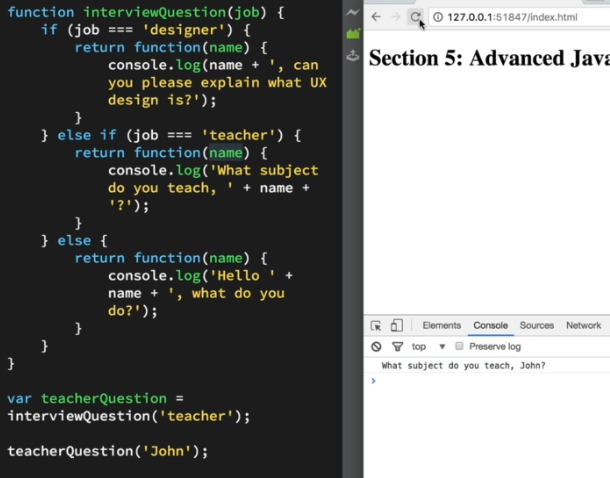
1. Execution phase: Where the saved functions and commands are executed.

**Primitives Vs Objects**

Primitive data types own their own memory location, hence if passed as an argument, a makes a copy of age and hence changing it doesn’t change value of age variable. However, objects when copied, point to same location and hence have a shallow copying, resulting in updation of both when ever one is updated.

**So, if a copy of object is changed, the copied object also changes.**

**Functions are also objects, hence a function can return another function:**

****

*This can be used instead of making a variable and then calling it out :* ****

**Creating object :**

Usually if we are not going to use inheritance or multiple objects of similar kinds we define objects directly like this:

var person = {

firstName : "John",

lastName : "Doe",

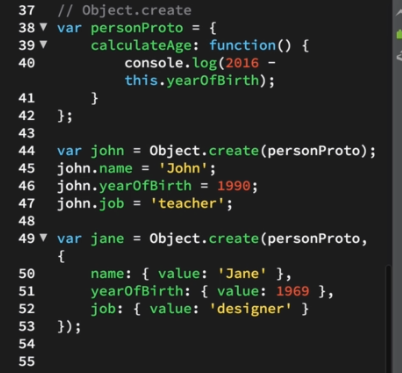
age : 50,

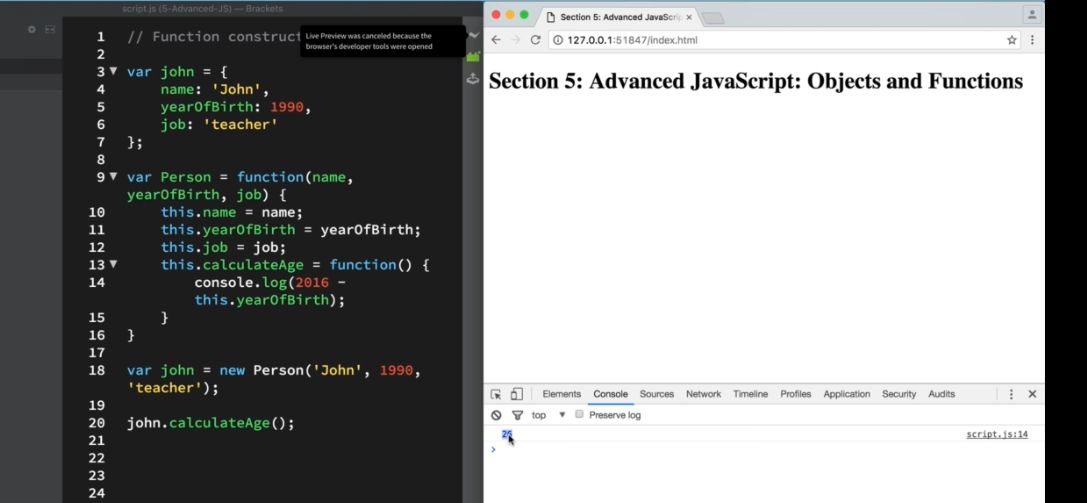
eyeColor : "blue"

};

But if we want to make multiple instances we can either use function or object.create() method to make a new object.

USING FUNCTION: USING OBJECT.CREATE() :

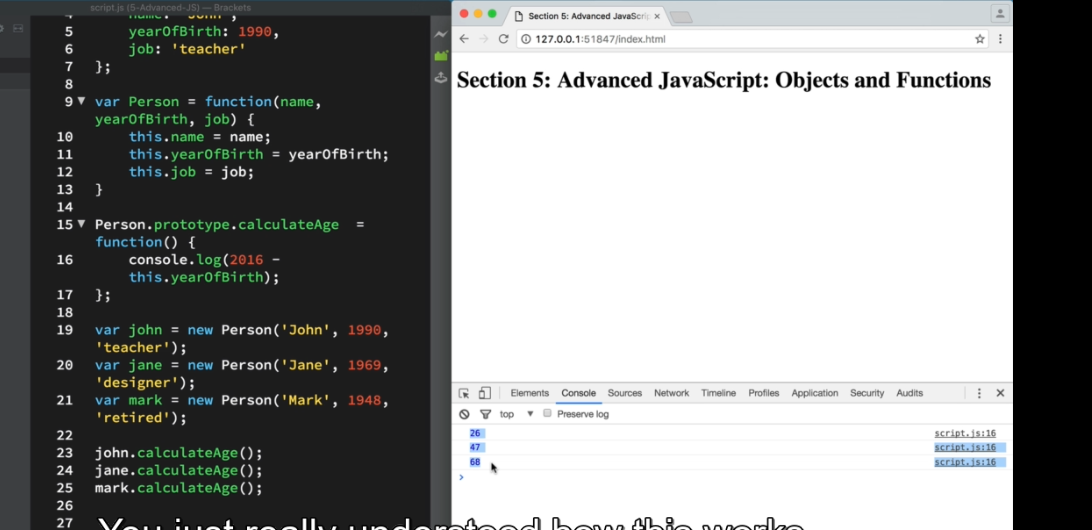




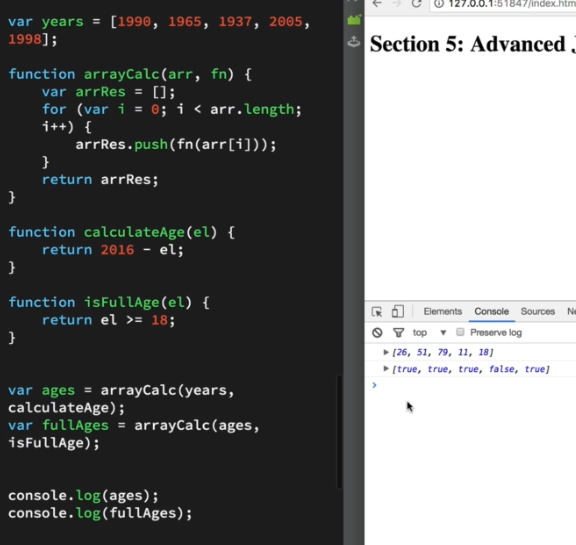
Object.create helps us make a new object out of an existing object.

A more descriptive and clarified version here on [stackOverflow](https://stackoverflow.com/questions/4166616/understanding-the-difference-between-object-create-and-new-somefunction)

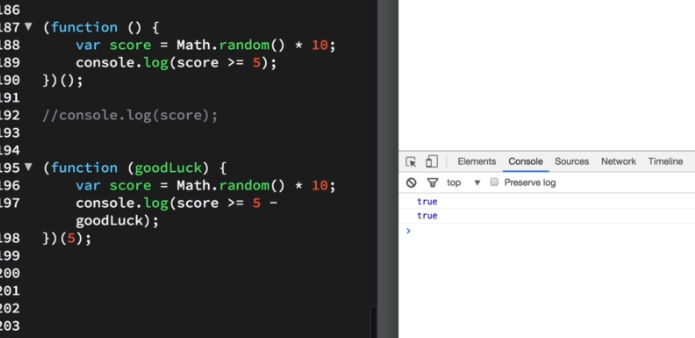
Adding a function in prototype



**Functions can be passed as arguments:**

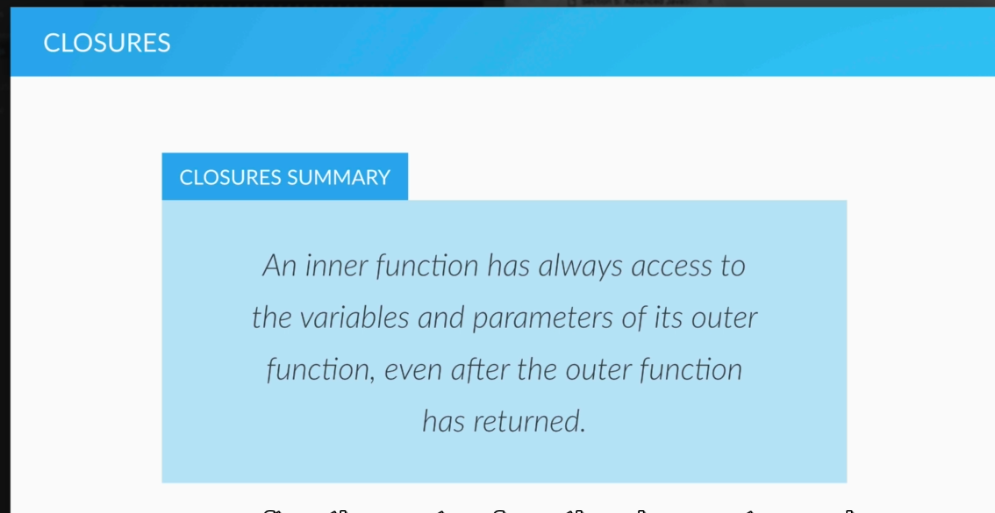
****

**Immediately invoked function expressions (IIFE) – where functions are called right where they are defined.**

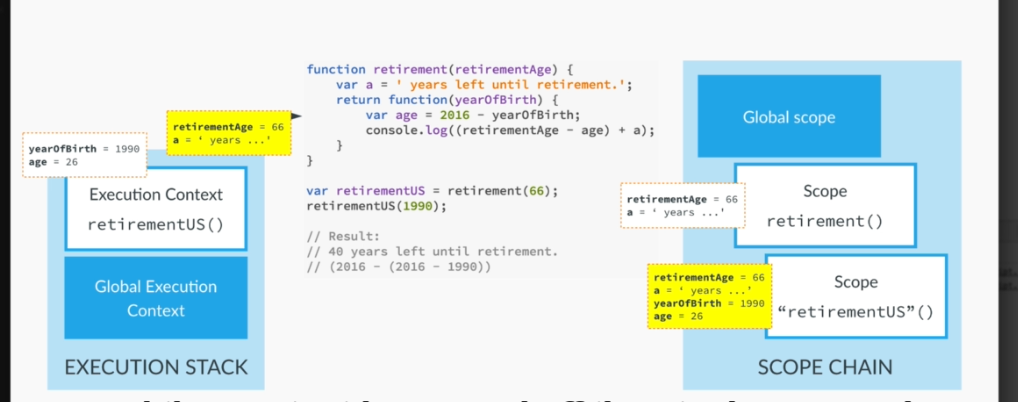
****

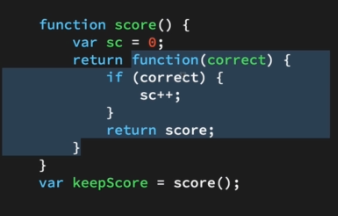
The primary reason to **use** an **IIFE** is to obtain data privacy. Because JavaScript's var scopes variables to their containing function, any variables declared within the **IIFE** cannot be accessed by the outside world.

**Closure**

****

**Even if retirement execution context is called off, its variable stays and hence can be used by the inner function.**

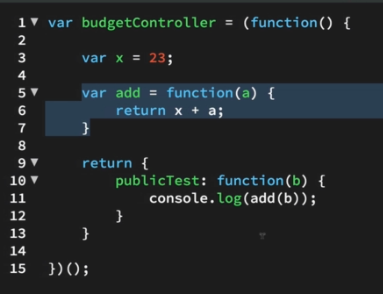
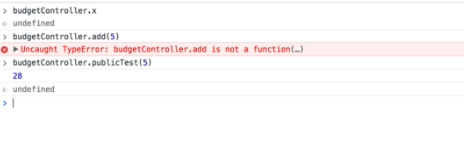
****

**Hence closure can be used to store variables and keep updating them.  
**

Here keepscore is a function which can access sc and keep incrementing it and saving it as well.

**Closure and IIFEs can be used together to make isolated modules whose access can be controlled.**

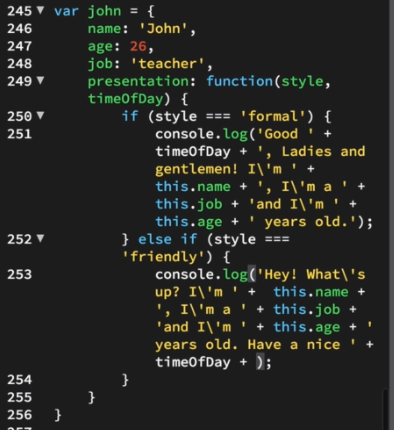
* **IIFE gives us privacy**
* **Closure let’s us give access of things that we want to.**

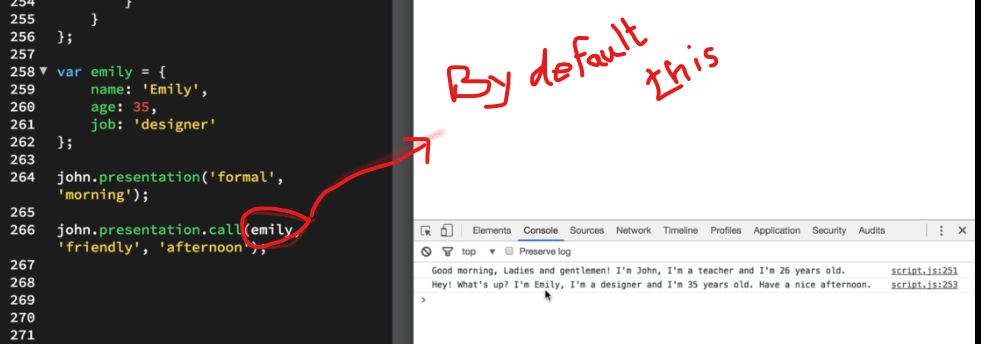
****

Hence x and add() cannot be accessed outside (due to IFFE) but public test can be used to make use of x and add() (due to closure)

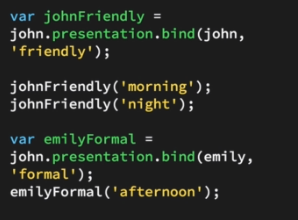
**Method borrowing :**

We can use .call() to borrow another object’s function.

****

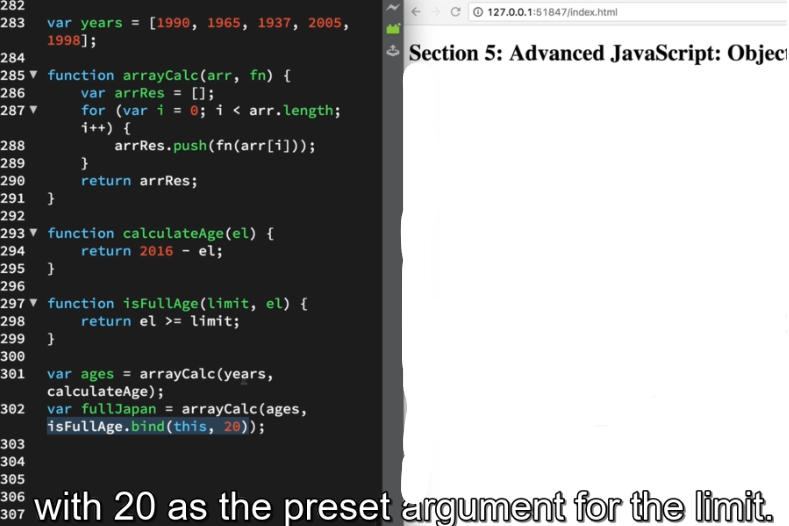
****

**Bind :**

****

Basically binding the function with scope and an argument.

**Beauty of bind eg:**

****

**FAQs:**

Why to use add event listener for click instead of using onclick attrb in the html file?

Coz few browsers doesn’t identify onclick. Moreover onclick overrides a prev onclick command.