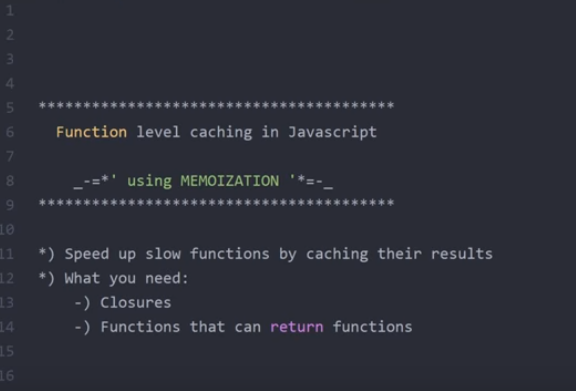
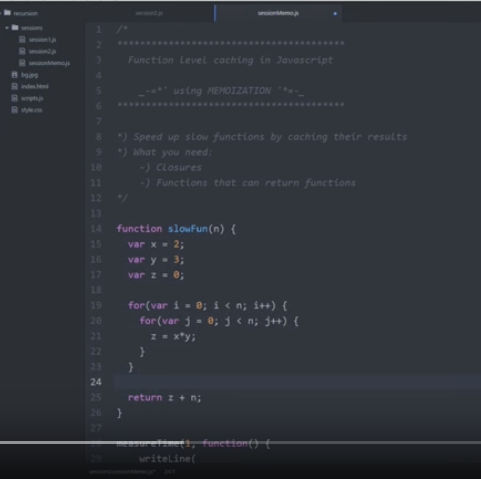
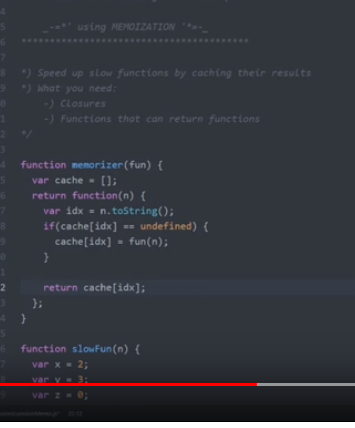
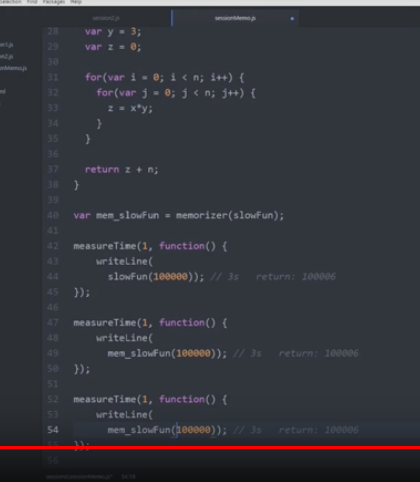
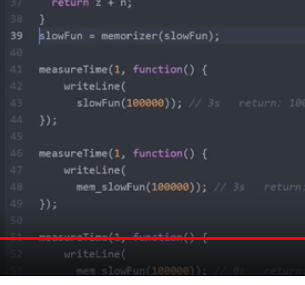
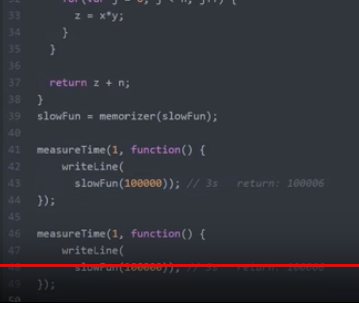
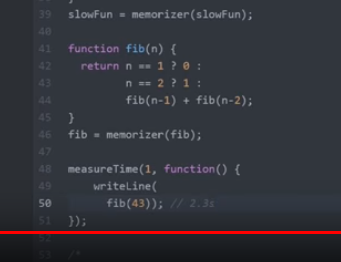
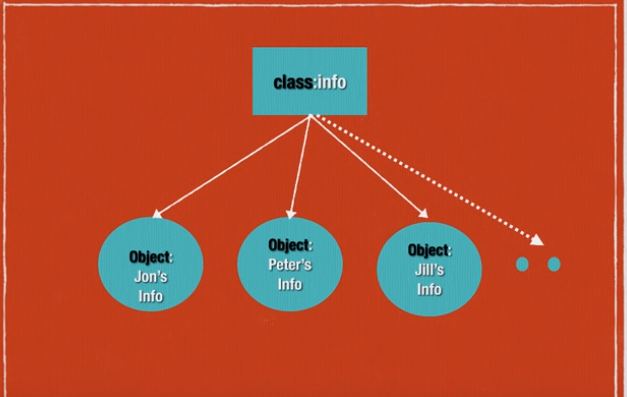
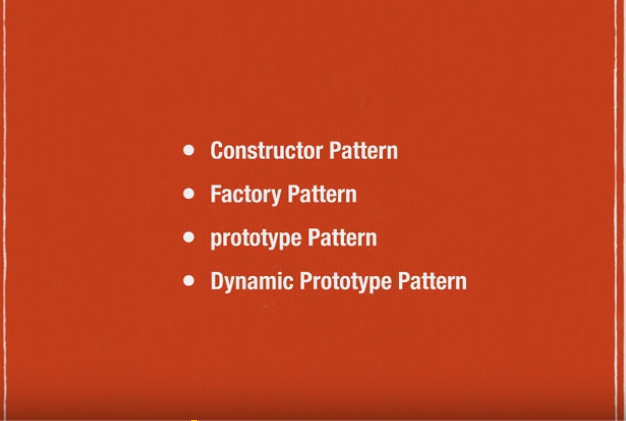
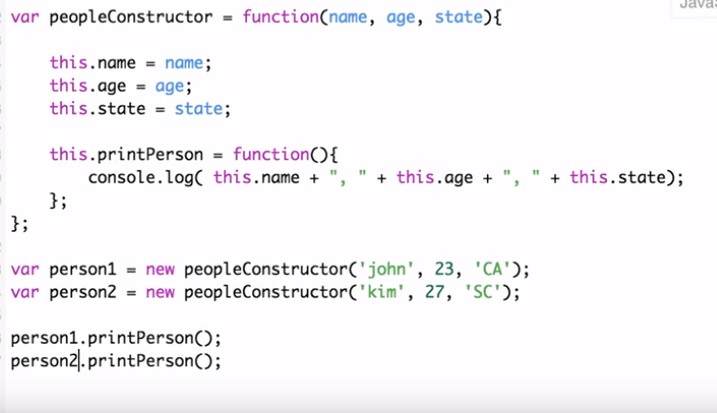
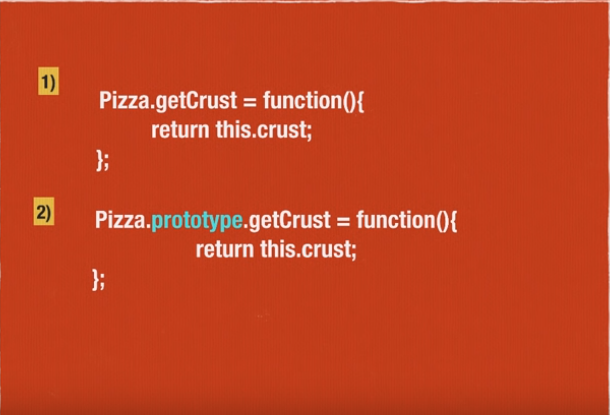
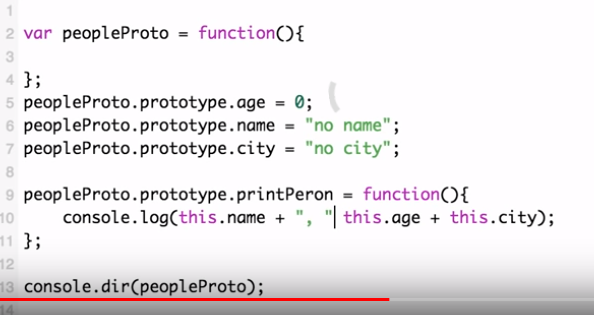
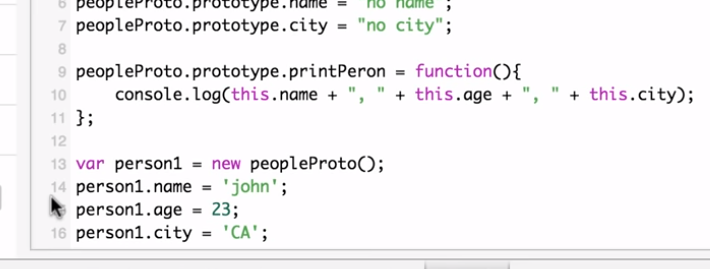
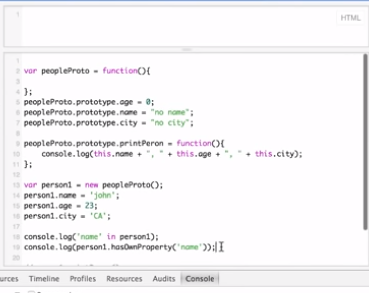
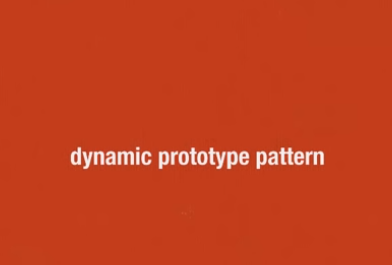
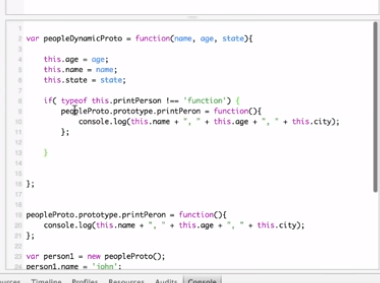
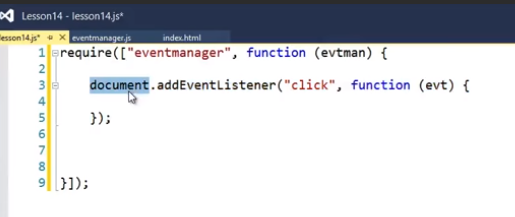
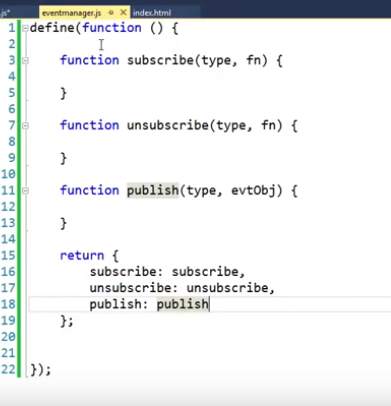
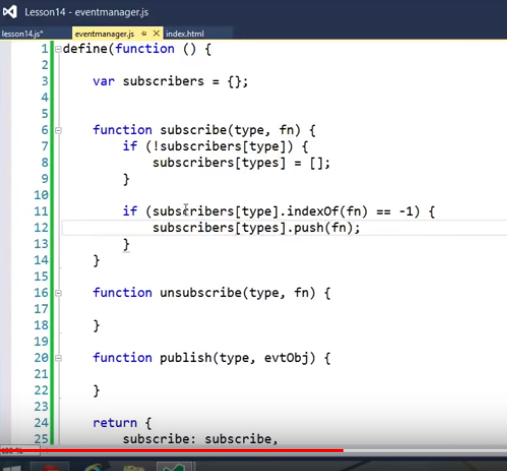
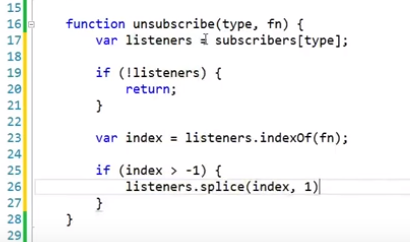
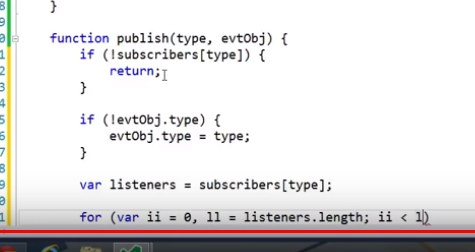
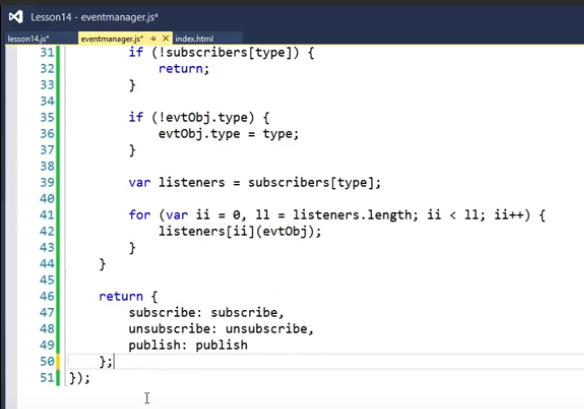
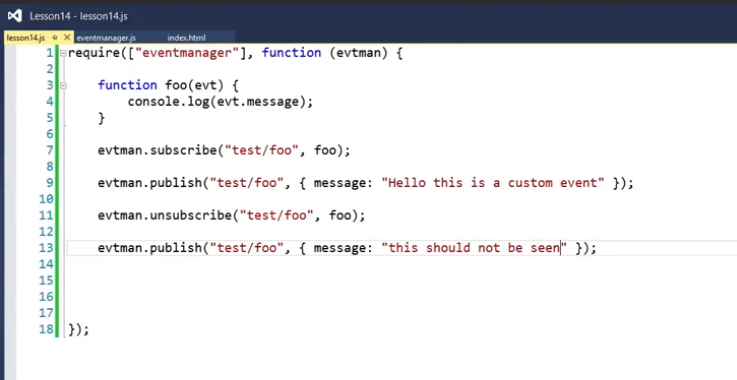
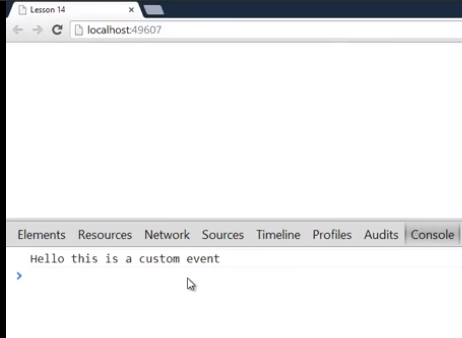
**Question 1 - Cache and Memorization in JavaScript**

1. 
2. 
3. 
4. 
5. 
6. 
7. We can override the definition of slowFun with the memorizer of the slowFun.
8. On line 39 we are changing the behavior of the function slowFunc.
9. 
10. Now, we will write a function which will calculate f(n for Fibonacci number).
11. 

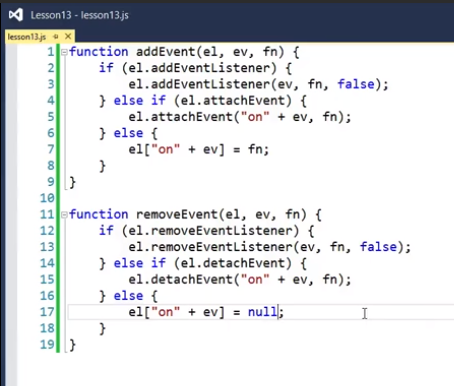
**Question 2: Design Patterns in JavaScript and Drawbacks of the design pattern**

1. Factory Pattern
2. Constructor Pattern
3. Prototype Pattern
4. Above are Object creation patterns
5. 
6. 
7. Factory Pattern
8. Functions behave like classes when we assign them to a variable
9. 
10. Above function behaves like a factory
11. 
12. Now, we will look at constructor pattern
13. 
14. The problem with the constructor pattern is it brings redundancy in the form of this object, which is a new object, if we have some method on this and then we create 1000 objects – all the objects will have same function causing redundency.
15. Now, we will move to next Pattern – Prototype pattern
16. Here every object will get a shared space prototype
17. 2 ways to create method:
18. 
19. 
20. 
21. 
22. 
23. person1 is light weight
24. you cannot create everything in 1 line you have to create promise in first line first.
25. Next pattern:
26. 
27. 
28. 

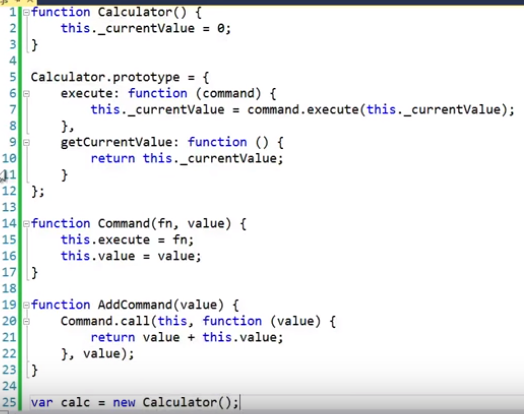
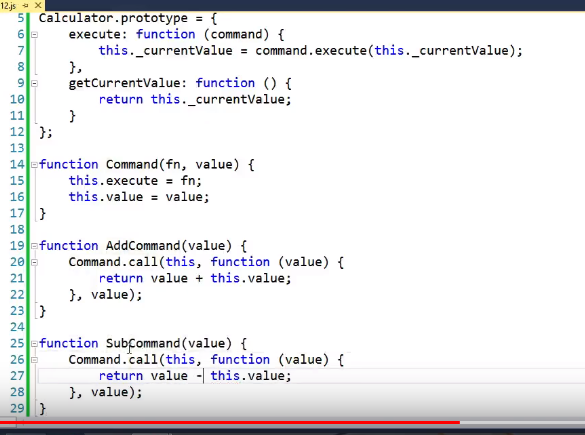
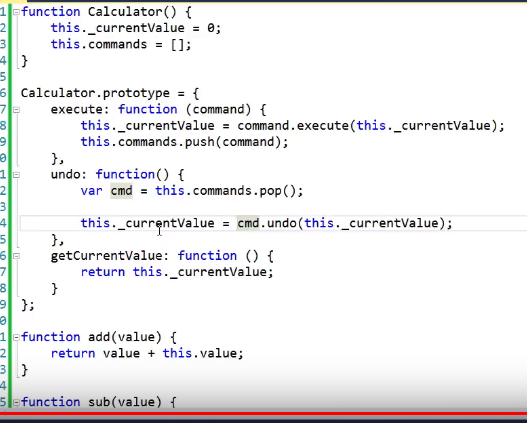
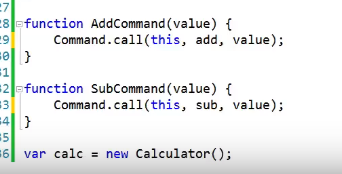
**Design Patterns**

1. <https://www.youtube.com/watch?v=HqPuodp6H-4&list=PLrzrNeNx3kNHsaPfrpPo0AlW-MhJE6gOA>
2. Patterns:
3. Some of Gang of 4 design patterns
4. 1. Behaviour Pattern
5. -**The Observer** - Subjects, observable - other objects are the observers
6. Javascript and DOM are higly event driven
7. -publisher-subscriber pattern
8. 
9. 
10. 
11. 
12. 
13. 
14. The require() **method** is used to load and cache JavaScript modules. So, if you want to load a local, relative JavaScript **module** into a **Node.js** application, you can simply use the require() **method**.
15. 
16. 

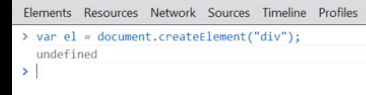
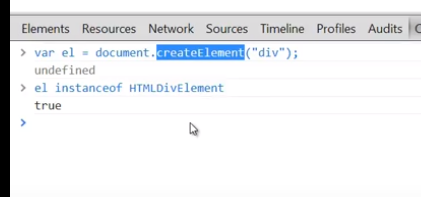
**Façade Pattern**

1. Fasade pattern is an important pattern in the web application development.
2. It hides the complexity between client and server communication.
3. We will write 2 functions here; 1 to add event handlers and other to remove event handlers.
4. We need 3 things in the function that we are creating element, event and the function that gets fired when the event is triggered.
5. 
6. We take complex code and hide it in easier to use API.

**Command Pattern**

1. It gives us the ability to separate the responsibility of issuing the command from anything that is executing the command. This flexibility allows us to write a clean and flexible API.
2. Command pattern seeks to use an object as a verb
3. Here we have written a simple calculator datatype.
4. 
5. 
6. 
7. 

**Factory Pattern**

1. Example of factory method:
2. 
3. 

**Singleton Pattern**

1. Singleton is to ensure that a class only has 1 instance and provide a global point of access to it.

**Decorator Pattern**