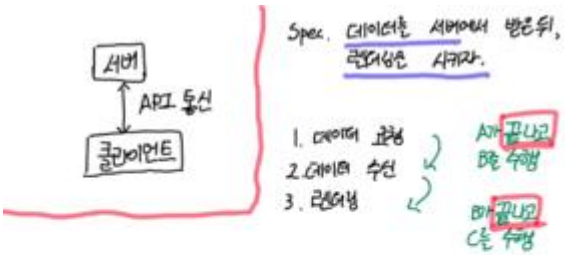
함수 3개

Ⓐ – A호출
Ⓑ – B호출
Ⓒ – C호출

한 땀 한 땀 처리한다.
(동기방식)
↳ 일단은 순서대로 동작
↳ A가 끝나야 B하고,
↳ B가 끝나야 C를 함

A()
B()
C()




Spec. 데이터는 서버에서 받은 뒤,
랜더링은 시키자.

1. 데이터 요청
2. 데이터 수신
3. 랜더링

A가 끝나고 B를 수행
B가 끝나고 C를 수행

JS의 비동기

〈 Promise 〉
↳ 특정한 상태를 저장할 수 있는 객체.

Promise
.then( callback )
↳ 여기서 Promise 에서 처리한 데이터 (=상태) 를 받아서 처리

예시) fetch ( " data.json` )
↳ 데이터를 불러와라
↳ 네트워크 속도에 따라 데이터 받아오는 속도가 다를 수 있음
↳ 그래서 then을 사용

fetch
Server
Res
then
Do Something
분석

.then ( )
↳ 데이터가 오면 OO을 해라
↳ 보통 .json() 에서 Data를 JSON화 시킴

JS는 모든것이 객체다
by Jhon Cho

Everything is Object

JS는 객체는 Heap에 저장
(힙)

JS가 함수는 실행할 때는
두가지 개념을 활용하여사용

Call Stack  Task Queue

함수를 호출했다고 하면,
일단 해당 함수는 Call Stack에 쌓임

만약 비동기 작업을 만나면,
JS Engine은 해당 작업은 Web API에 이관
(AJAx, DOM Events, Timer)


Web API가 비동기 작업을 끝내면,
해당 작업은 Task Queue에 넣음

Call Stack의 작업이 모두 해소되고,
Call Stack이 완전히 비면,
Task Queue의 아이템은 Call Stack에 넣음

JS가 Single thread인 이유

= Call Stack이 하나임

a()
b()
setTimeout ( c , 0 )
d()
e()

순서: ?

1 a
2 b
3 d
4 e
5 c

아무리 0초 딜레이 주더라도 Call Stack이 아니라
Task Queue에 있다가 동작해서 c가 가장 마지막