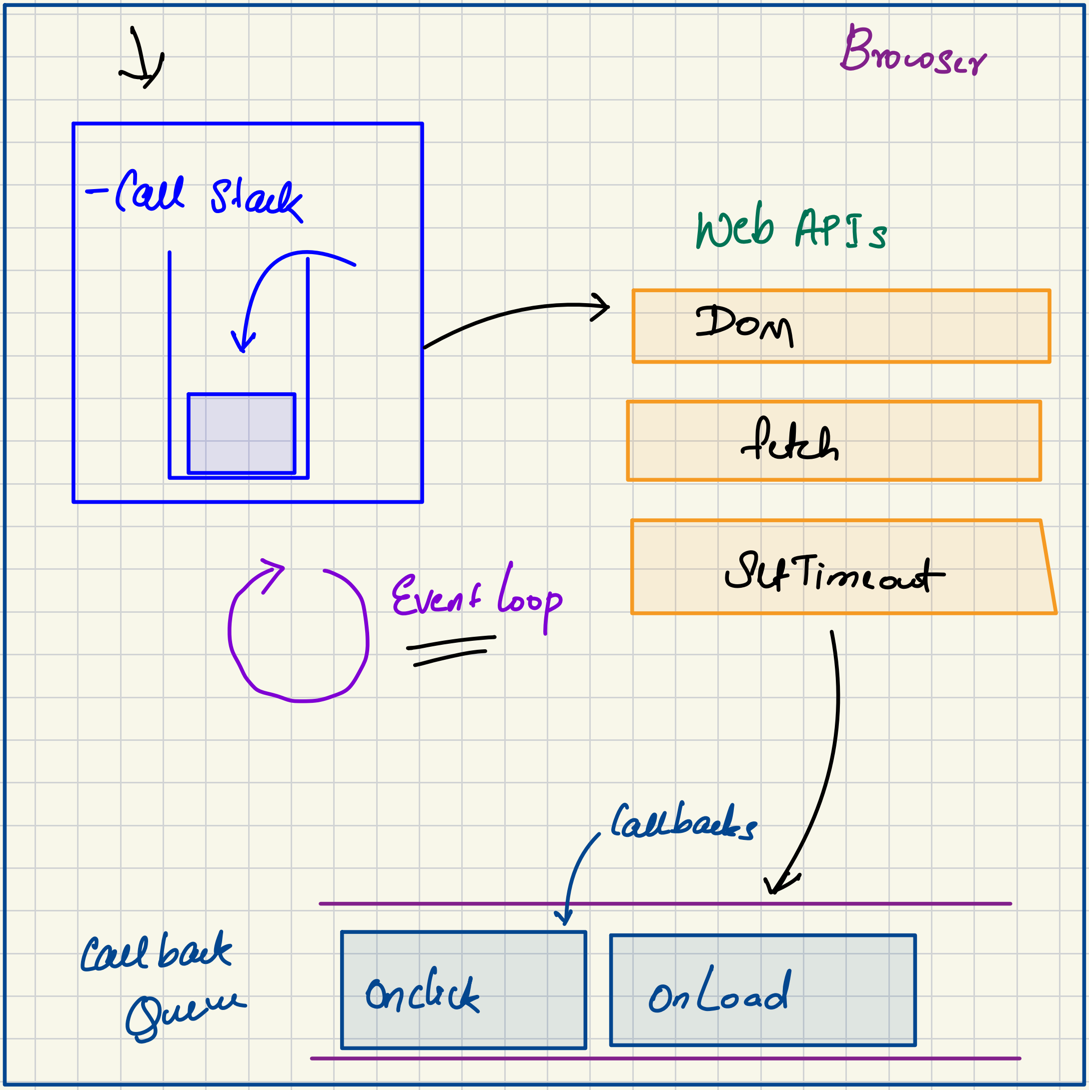
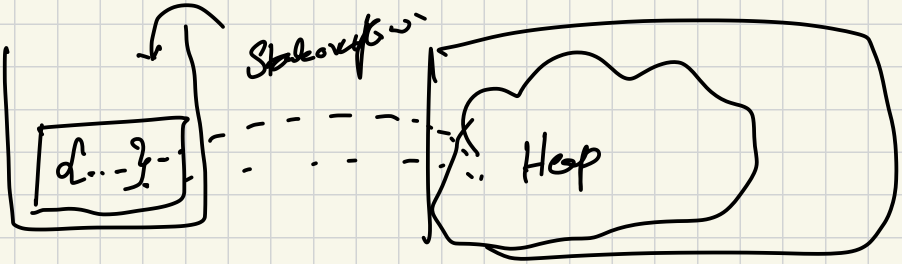
EVENT LOOP

Author:- Shrikanth .R-S

Components of the browser:





Heap is unstructured data, usually stuff like array, objects are stored here.

Stack → Anything that needs to be executed is executed from the call stack.
[This is covered in detail in another lecture]

Browser & web APIs -

Javascript is single threaded, synchronous by nature. The other capabilities are provided by the Browser & WebAPIs

Such as SetTimeout, Geolocation, Console...

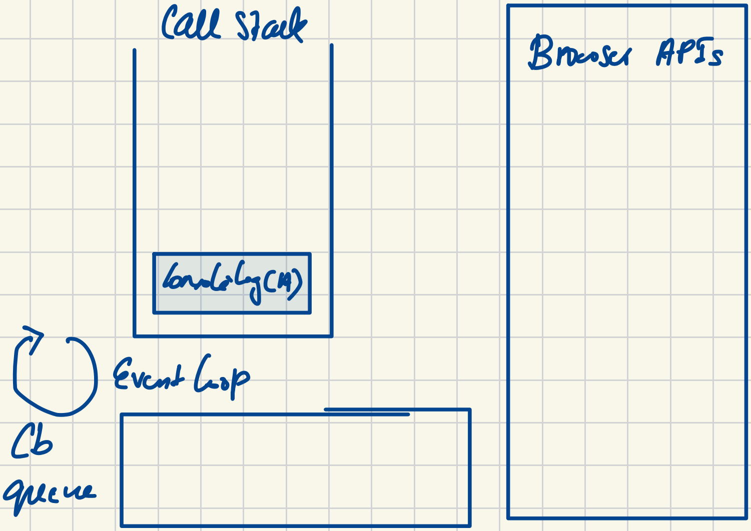
Lets understand Eventloop with an example:

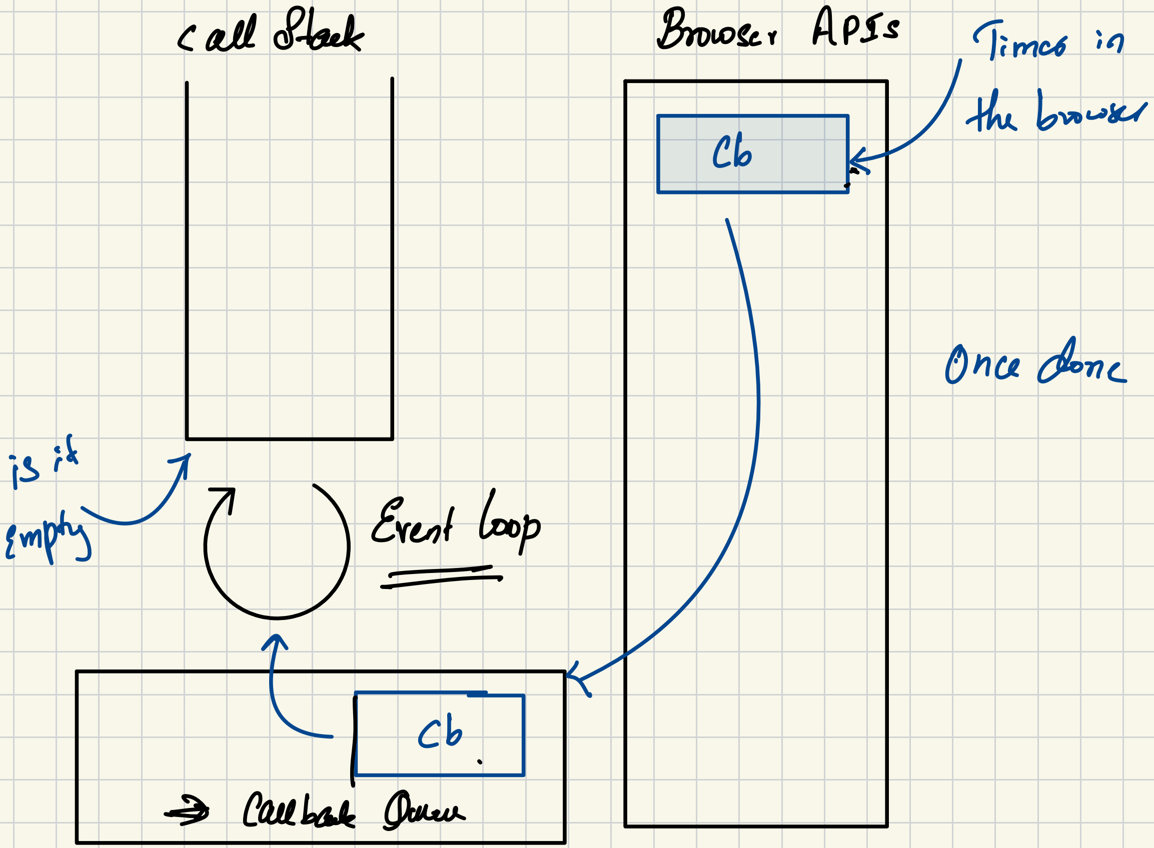
0 → 1000
100

```
function root() {
        console.log('A');
     → SetTimeout(() => { console.log('B'); }, 0);
    →  console.log('C)');
}
```

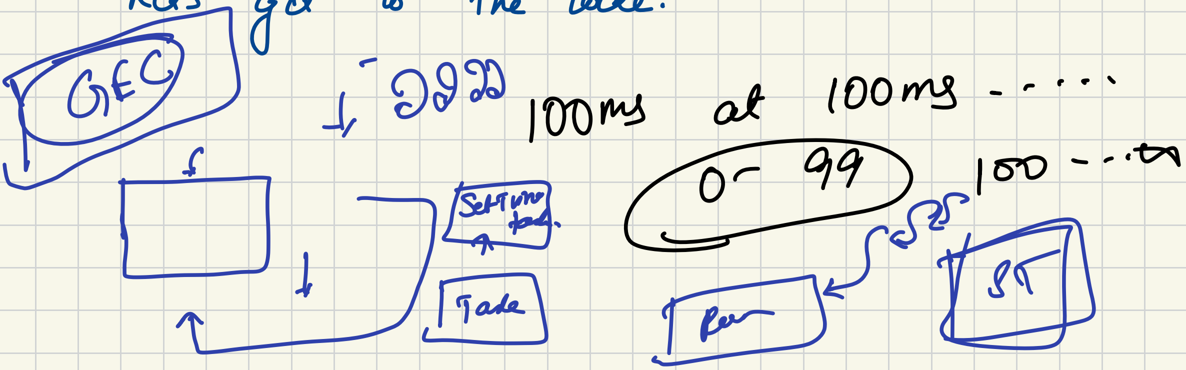
Output is A, C, B.

→ Lets understand it a bit more.

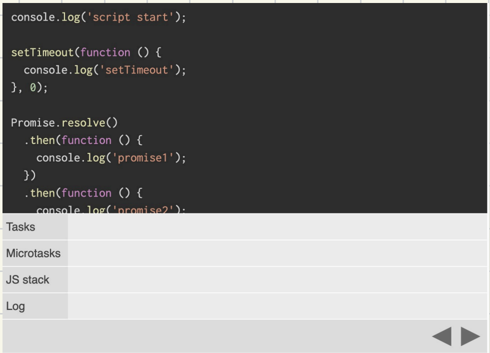




Lets get to the code.



microtask Queue

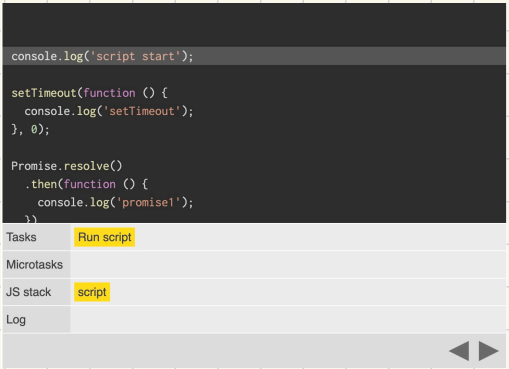


```
console.log('script start');

setTimeout(function () {
  console.log('setTimeout');
}, 0);

Promise.resolve()
  .then(function () {
    console.log('promise1');
  })
  .then(function () {
    console.log('promise2');
```

Tasks	
Microtasks	
JS stack	
Log	



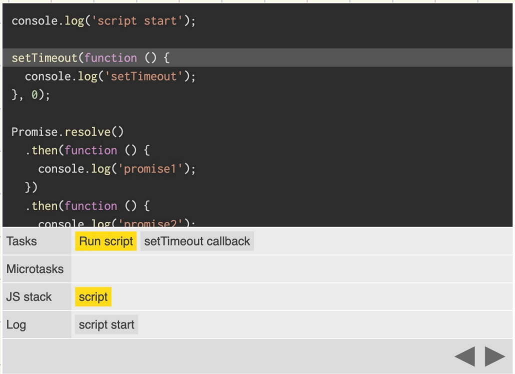
```

console.log('script start');

setTimeout(function () {
  console.log('setTimeout');
}, 0);

Promise.resolve()
  .then(function () {
    console.log('promise1');
  })
```

Tasks	Run script
Microtasks	
JS stack	script
Log	

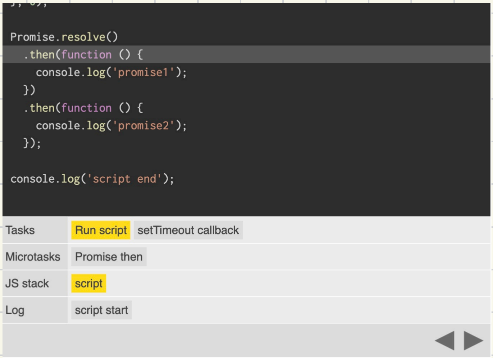


```
console.log('script start');

setTimeout(function () {
  console.log('setTimeout');
}, 0);

Promise.resolve()
  .then(function () {
    console.log('promise1');
  })
  .then(function () {
    console.log('promise2');
```

Tasks	Run script	setTimeout callback
Microtasks		
JS stack	script	
Log	script start	

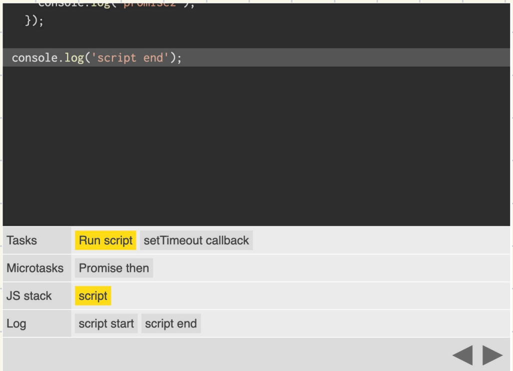


```
}, 0);

Promise.resolve()
  .then(function () {
    console.log('promise1');
  })
  .then(function () {
    console.log('promise2');
  });

console.log('script end');
```

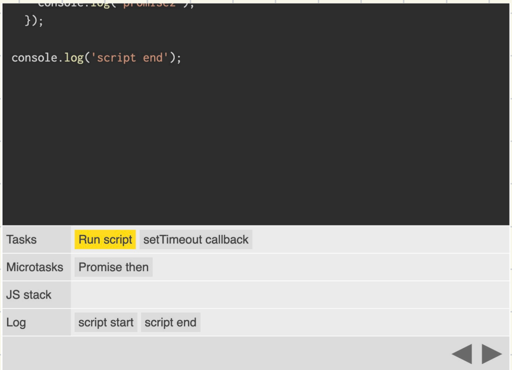
Tasks	Run script	setTimeout callback
Microtasks	Promise then	
JS stack	script	
Log	script start	



```
    console.log('promise2');
  });

console.log('script end');
```

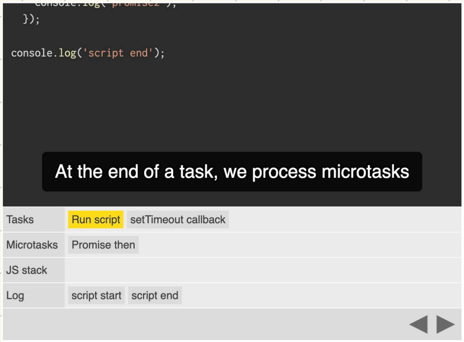
Tasks	Run script	setTimeout callback
Microtasks	Promise then	
JS stack	script	
Log	script start	script end



```
    console.log('promise2');
  });

console.log('script end');
```

Tasks	Run script	setTimeout callback
Microtasks	Promise then	
JS stack		
Log	script start	script end



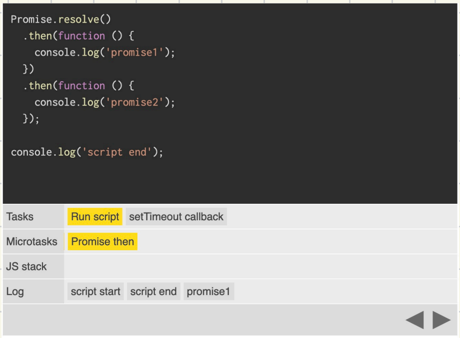
```
    console.log('promise2');
  });

console.log('script end');
```

At the end of a task, we process microtasks

Tasks	Run script setTimeout callback
Microtasks	Promise then
JS stack	
Log	script start script end

◀ ▶

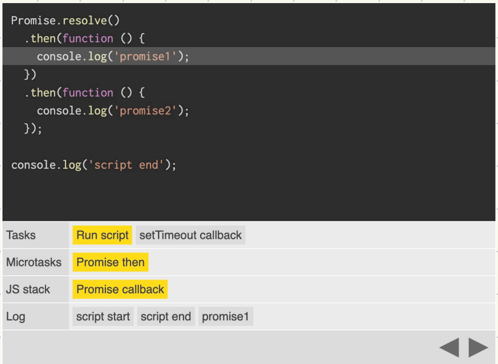


```
Promise.resolve()
  .then(function () {
    console.log('promise1');
  })
  .then(function () {
    console.log('promise2');
  });

console.log('script end');
```

Tasks	Run script setTimeout callback
Microtasks	Promise then
JS stack	
Log	script start script end promise1

◀ ▶

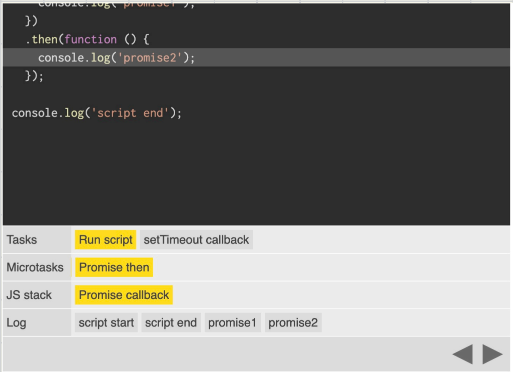


```
Promise.resolve()
  .then(function () {
    console.log('promise1');
  })
  .then(function () {
    console.log('promise2');
  });

console.log('script end');
```

Tasks	Run script setTimeout callback
Microtasks	Promise then
JS stack	Promise callback
Log	script start script end promise1

◀ ▶

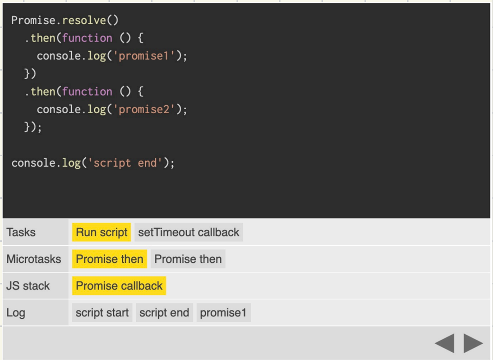


```
    console.log('promise1');
  })
  .then(function () {
    console.log('promise2');
  });

console.log('script end');
```

Tasks	Run script setTimeout callback
Microtasks	Promise then
JS stack	Promise callback
Log	script start script end promise1 promise2

◀ ▶

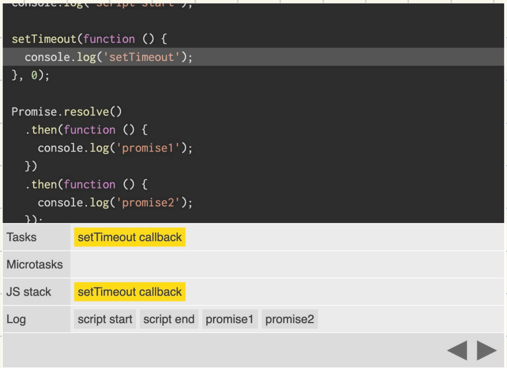


```
Promise.resolve()
  .then(function () {
    console.log('promise1');
  })
  .then(function () {
    console.log('promise2');
  });

console.log('script end');
```

Tasks	Run script setTimeout callback
Microtasks	Promise then Promise then
JS stack	Promise callback
Log	script start script end promise1

◀ ▶



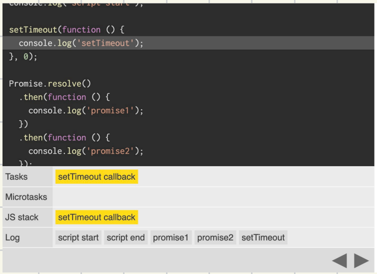
```
console.log('script start');

setTimeout(function () {
  console.log('setTimeout');
}, 0);

Promise.resolve()
  .then(function () {
    console.log('promise1');
  })
  .then(function () {
    console.log('promise2');
  });
```

Tasks	setTimeout callback
Microtasks	
JS stack	setTimeout callback
Log	script start script end promise1 promise2

◀ ▶



```
console.log('script start');

setTimeout(function () {
  console.log('setTimeout');
}, 0);

Promise.resolve()
  .then(function () {
    console.log('promise1');
  })
  .then(function () {
    console.log('promise2');
  });
```

Tasks	setTimeout callback
Microtasks	
JS stack	setTimeout callback
Log	script start script end promise1 promise2 setTimeout

◀ ▶