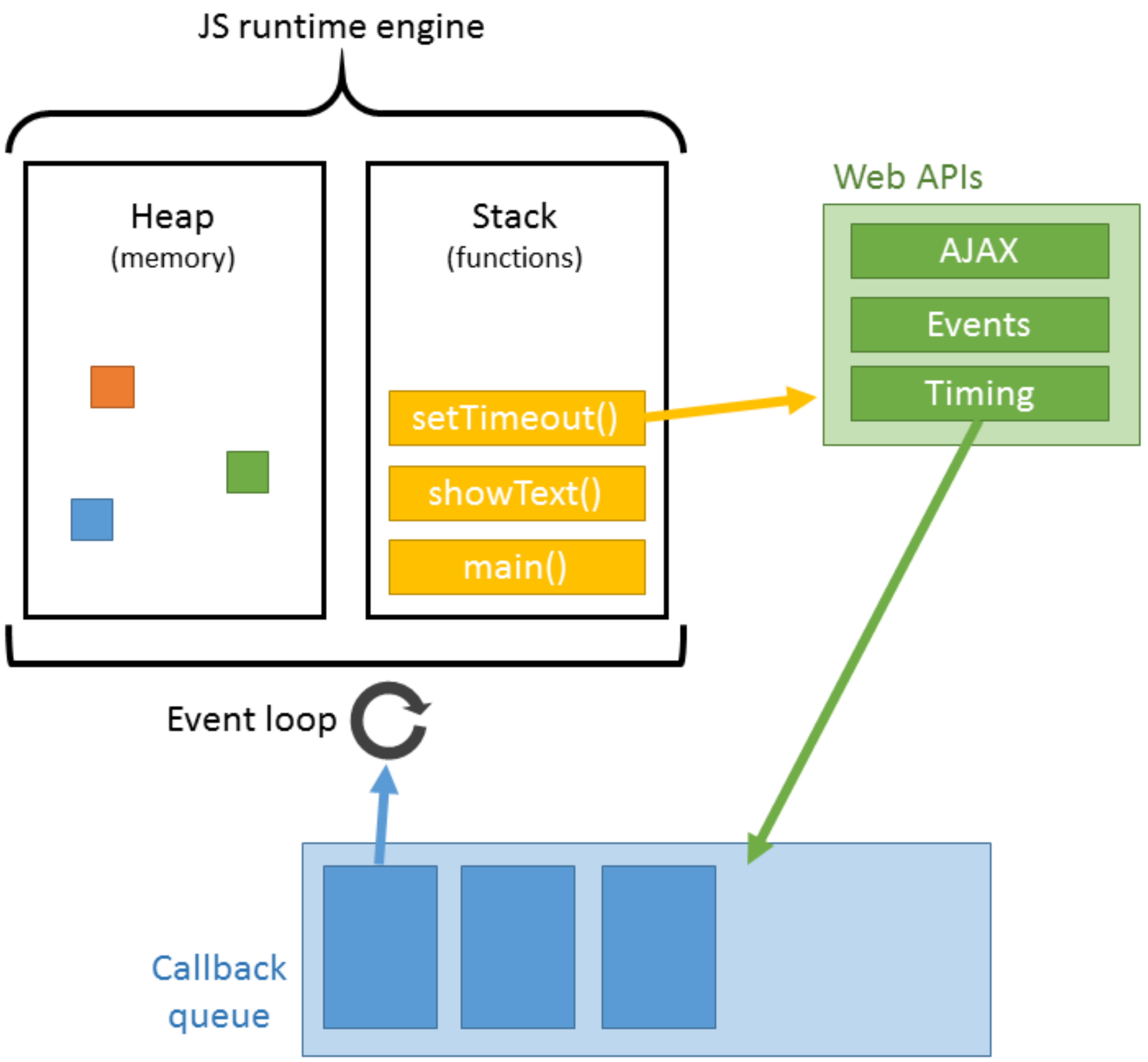
THE JAVASCRIPT EVENT LOOP

OBJECTIVES

- DESCRIBE HOW THE CALL STACK, EVENT QUEUE AND EVENT LOOP WORK TOGETHER TO EXECUTE CODE OR DELAY CODE EXECUTION
- DESCRIBE THE DIFFERENCE BETWEEN "SYNCHRONOUS" & "ASYNCHRONOUS" CODE
- USE SETTIMEOUT TO DELAY THE EXCUTION OF CODE

IN WHAT ORDER WILL THIS CODE RUN?

```
setTimeout(() => console.log('first'), 1000)
console.log('second')
```


JS runtime engine
Heap
(memory)
Stack
(functions)
setTimeout()
showText()
main()
Web APIs
AJAX
Events
Timing
Event loop
Callback
queue

TERMS

- HEAP
- STACK
- QUEUE
- EVENT LOOP

DEFINITION

- HEAP - MEMORY WHERE FUNCTION ARE STORED

DEFINITION

- STACK - FIRST IN, LAST OUT LIST OF FUNCTIONS TO PERFORM.

  AKA: CALL STACK

DEFINITION

- QUEUE - FIRST IN, FIRST OUT LIST OF FUNCTIONS TO ADD TO THE CALL STACK ONCE THE CALL STACK IS EMPTY.

AKA: EVENT QUEUE, CALLBACK QUEUE, MESSAGE QUEUE

DEFINITION

- EVENT LOOP - A LOOP THAT PROCESSES MESSAGES IN THE QUEUE AND MOVES THEM FROM THE QUEUE TO THE STACK

SYNCHRONOUS CODE

IN SYNCHRONOUS PROGRAMS, IF YOU HAVE TWO LINES OF CODE (L1 FOLLOWED BY L2), THEN L2 CANNOT BEGIN RUNNING UNTIL L1 HAS FINISHED EXECUTING.

DIAGRAM STACK/QUEUE FOR SYNCHRONOUS CODE

```
function multiply(a, b) { return a * b }

function square(n) { return multiply(n, n) }

function printSquare(n) {
  var squared = square(n)
  console.log(squared)
}

printSquare(4)
```

ASYNCHRONOUS CODE

IN ASYNCHRONOUS PROGRAMS, YOU CAN HAVE TWO LINES OF CODE (L1 FOLLOWED BY L2), WHERE L1 SCHEDULES SOME TASK TO BE RUN IN THE FUTURE, BUT L2 RUNS BEFORE THAT TASK COMPLETES.

DIAGRAM STACK/QUEUE FOR ANSYNCHRONOUS CODE

```
console.log("1: Beginning")

setTimeout(() => { console.log("2: First timeout") }, 1000)

console.log("3: Middle")

setTimeout(() => { console.log("4: Second timeout") }, 1000)

console.log("5: End")
```

AN ASIDE … WHAT IS A STACK OVERFLOW?

```
function oopsIDidItAgain() {
  return oopsIDidItAgain()
}

oopsIDidItAgain()
```

IN WHAT ORDER WILL THIS CODE RUN? NOTE THE TIMEOUT IS ZERO

```
setTimeout(() => console.log('cat'), 0)
console.log('dog')
```

IF GETDATA IS ASYNCHRONOUS, WHY WON'T THIS CODE WORK?

```
function getData() {
  var data;
  $.get("example.php", function(response) {
    data = response;
  });
  return data;
}

var data = getData();
console.log("The data is: " + data);
```

DOES THE FOLLOWING CODE TAKE ABOUT 1 SECOND OR 5 SECONDS?

```
setTimeout(() => { console.log("1: First timeout") }, 1000)
setTimeout(() => { console.log("2: Second timeout") }, 1000)
setTimeout(() => { console.log("3: Third timeout") }, 1000)
setTimeout(() => { console.log("4: Fourth timeout") }, 1000)
setTimeout(() => { console.log("5: Fifth timeout") }, 1000)
```

RESOURCES

- HTTPS://WWW.YOUTUBE.COM/WATCH?V=8AGHZQKOFBQ
- HTTP://LATENTFLIP.COM/LOUPE/
- HTTPS://DEVELOPER.MOZILLA.ORG/EN-US/DOCS/WEB/JAVASCRIPT/EVENTLOOP
- HTTPS://HACKERNOON.COM/UNDERSTANDING-JS-THE-EVENT-LOOP-959BEAE3AC40
- HTTPS://BLOG.CARBONFIVE.COM/2013/10/27/THE-