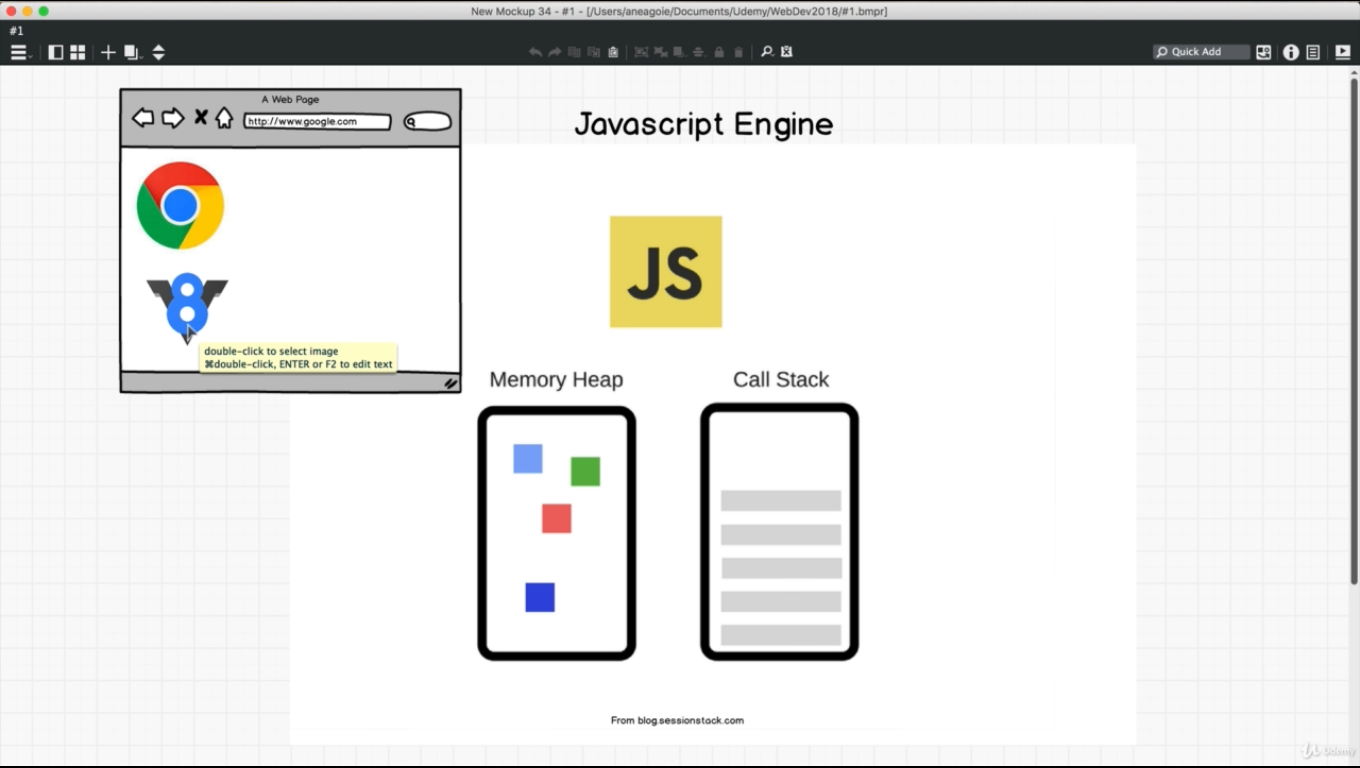
3. How JavaScript Works

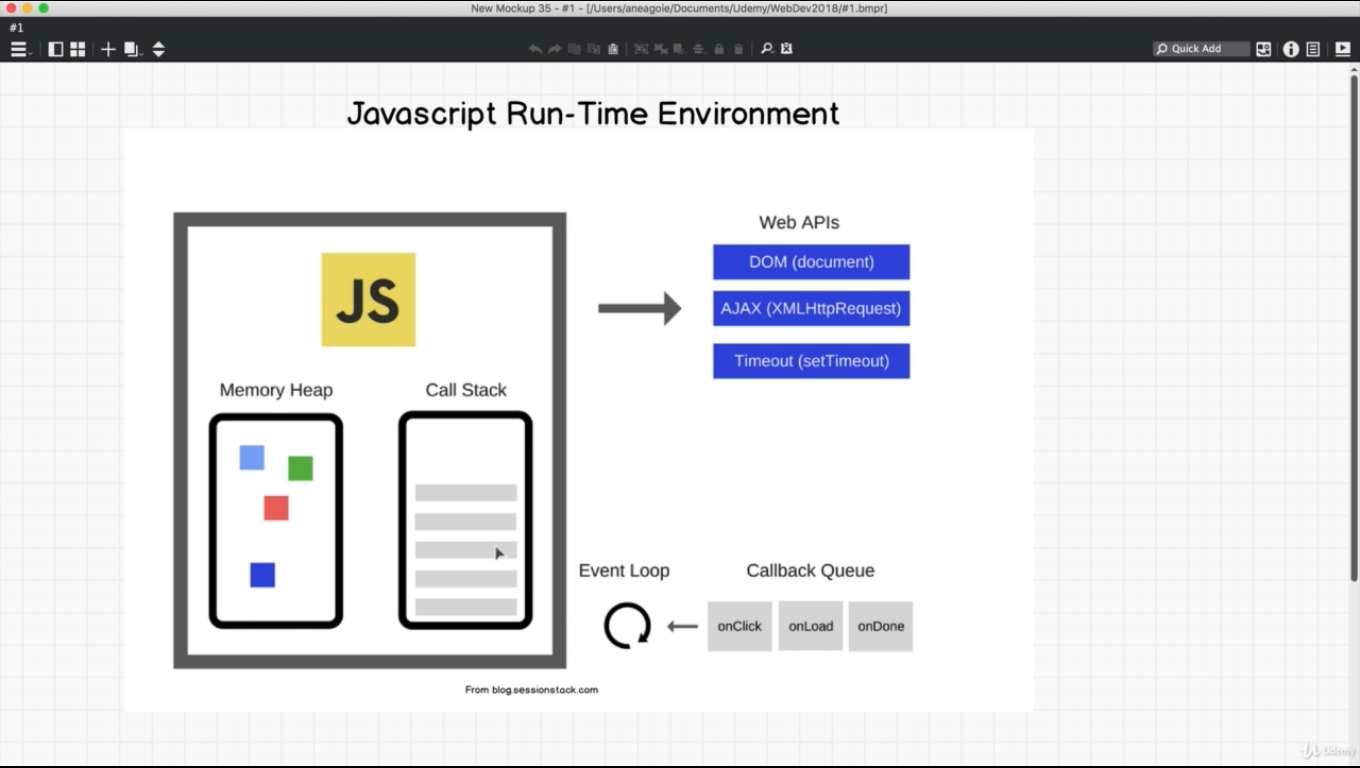
* What is a program?
  + Allocate memory.
  + Parse and execute.

Memory:

* Memory leak happens when unused variables are laying around and simply consumes memory. (akaimma).
* Global variables are bad for this reason. We forget to clean up after using it. It fills up the memory heap.



* JS is a single threaded language that can be non-blocking.
  + It has only one call stack.
* Why single threaded?
  + We don’t have to deal with complicated scenarios that arise in multi-threaded language.
  + Deadlock happens.
* Synchronous programming:
  + It executes one statement at a time.
  + Letter statement runs only when we finished running the first statement.
* Stackoverflow will occur if we have recursive call that fills up the stack resulting a stackoverflow.



* Javascript run-time environment: Comes with the browser, it has web API, event loop, callback Queue along with memory heap and call stack.
* To accomplish asynchronous programming in JS, we can borrow help from webapi.

INTERVIEW:

Whole process setTimeout ex:

* setTimeout goes to the stack.
* setTimeout triggers web API. Web API starts timer.
* Web API then checks what’s inside and sends it to callback queue.
* Event loop always checks if the call stack is empty. If it finds it empty then it checks the callback queue and puts whatever inside of that callback queue into the stack.
* Event listener is a callback because it listens or waits for an event to take place and only then it runs.

JS is a single threaded language that can be non-blocking. (asynchronous behavior)

* JS can be asynchronous with the help of callbacks, these callbacks run in the background through the callback queue, event loop and back to call stack.

4. Promises

* Promises are new feature in ES6.
* We have promises instead of callbacks.

Promises: A promise is an object that may produce a single value sometime in the future either a resolved value, or a reason that it’s not resolved (rejected).

3 states of promises:

* Fulfilled.
* Rejected.
* Pending.

Callback: When something is done execute this code.

* A callback has a pyramid of dom. Long nested code.
* .catch only runs only if something fails. .catch catches anything that is before it.

5. ES8 - Async Await

* Benefit of async await is that is has syntactical sugar for promise. Underneath the hood it is actually promise.
* Asynchronous code look synchronous.
* Promise is like IOU. It will return something in the future.
* We can put the await keyword in front of any fn that returns a promise.
* Instead of chaining we can now assign promises to a variable.
* Fetch fn is a promise.

JSON:

* A common use of JSON is to exchange data to/from a web server.
* When receiving data from a web server, the data is always a string.
* Parse the data with JSON.parse() , and the data becomes a JavaScript object.