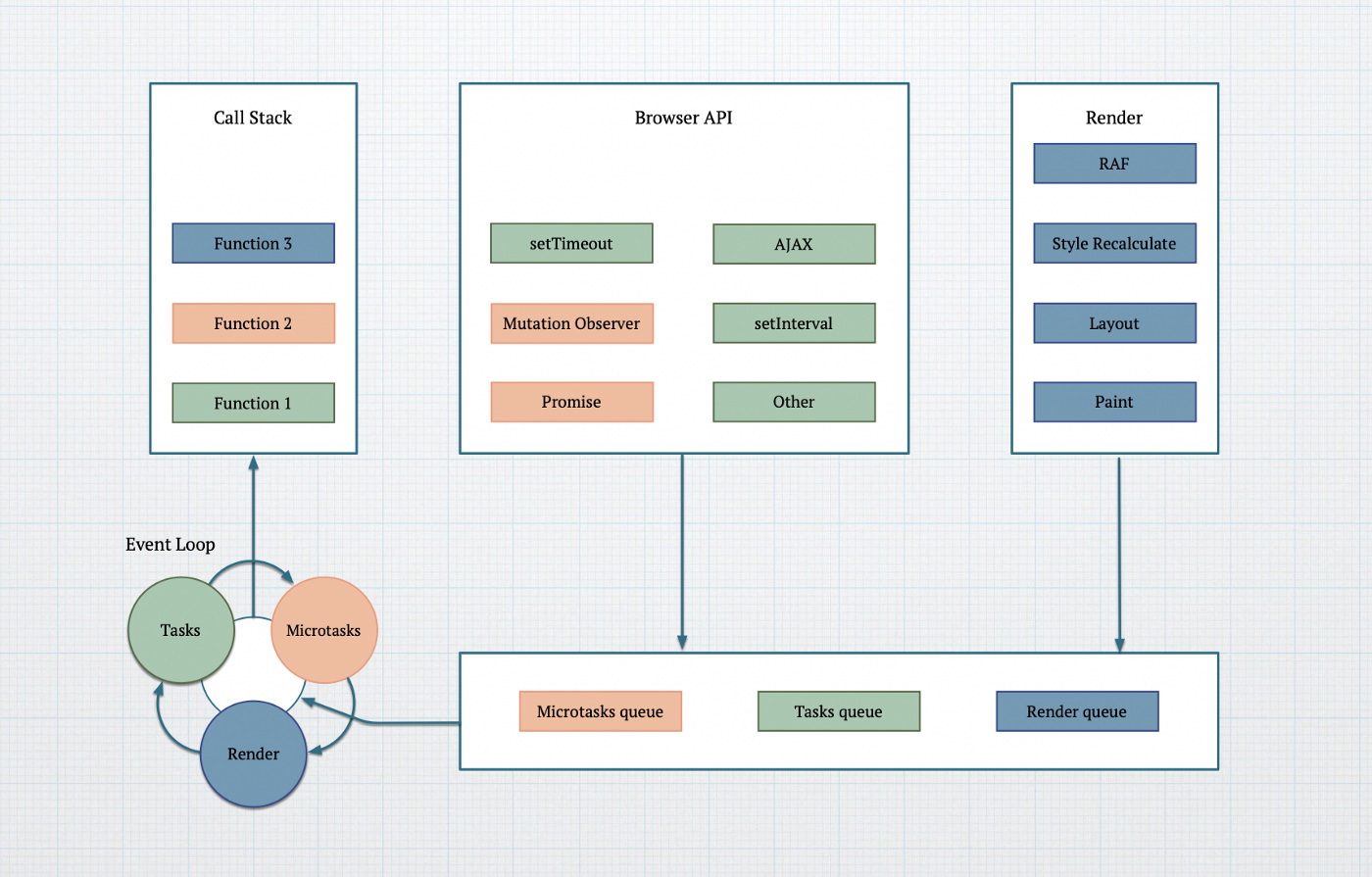
**Main Thread Loop**



From <https://docs.google.com/document/d/1GSHp-KoP4M1oq-vQaw8W8g93lm_k1zsykP_GAzRrbYo/edit#heading=h.fkrfg3qeppr8>

V8 is a main thread task.

## Executor, Microtask

Executor (From <https://javascript.info/promise-basics>:

“The function passed to the new Promise is called the executor. When a new Promise is created, the executor runs automatically. ”

Executor 1:

let promise = new Promise(function(resolve, reject) {

// **Executor**

});

const ret1 = promise.then((value) => {

// **Microtask**

console.log("Microtask1 # posted by Resolve: "+value);

return true;

});

Executor 2:

async function asyncCall() {

// **Executor**

await something();

// **Microtask**

}

{

await asyncCall();

}

## Queue a microtask, execute a microtask

**Who queues a microTask?** Other thread(Network), Main Thread(Call Promise.resolve.)

Main thread queues a microtask: time\_promisethen.js

const promise1 = new Promise((resolve, reject) => {

console.log("Main # Post MicroTask1 # In Promise new ");

**// Resolve posts a microtask**

resolve('Success From Promise!');

});

console.log('Main # after new promise');

const ret1 = promise1.then((value) => {

**// Microtask**

console.log("Microtask1 # posted by Resolve: "+value);

return true;

});

Other thread queues a microTask: time\_promisethen2await\_fetch.js

async function fetchData() {

const response = await fetch('./index.html');

// 1. Main thread **post task** to network (Done by fetch API);

// 2. When the network thread is done, **post microtask into** the main thread.

// **Microtask**

console.log("Microtask1 # posted by Chromium network");

return true;

}

**Who executes a microTask?** Main Thread (We don't talk about worker thread here).

## Three RULES

From <https://chromium.googlesource.com/chromium/src/+/HEAD/third_party/blink/renderer/platform/scheduler/TaskSchedulingInBlink.md> :

**#RULE 1** “At the moment Blink Scheduler treats tasks as an atomic unit — **if a task has started, it can’t be interrupted until it completes**. The scheduler can only choose a new task to run from the eligible tasks or can elect not to run any task at all.

”

“A [microtask](https://html.spec.whatwg.org/C#microtask) (which is a synonym of a task that happens to belong to a microtask queue) is run when

**#RULE 2**  (1) JavaScript is finished to run (JavaScript is also a Task, so JavaScript finished run equals “running task comes to an end”)

**#RULE 3** (2) the currently running task comes to an end. You can think of it as a hook that's invoked on a script or task completion.