Michael Abraha

Summary JS Engine

Let's start with how our JavaScript code is processed in Js engines?
When our code reaches the engine first it needs to be parsed. So, the Parser is responsible for tokenize the code and turn it to Abstract syntax tree (AST). After that the interpreter(ignition) comes next and start producing bytecode. So, in this area there jit profiler responsible for marking the statements as warm and Hot. A warm is a statement which get hit more than 100 and Hot is codes which are more than 100. So basically, it detects them send them to be compiled. The interpreter generates unoptimized bytecodes and are sent to optimizing compiler(turbofan). The compiler scans ahead of time and turns them to machine language . So, the optimizing compiler and generate highly optimized machine code.

There is a concept called the memory heap and stack. The memory heap a place in the memory where whenever a function is executed variables, arrays, objects …. Are allocated in the heap and the functions are placed at the execution context in the form of stack (LIFO). In the heap JavaScript has garbage collector whenever there are unused things that hold memory otherwise memory leak happens. And in the stack, there is certain amount of space in the execution context if exceeded we will have scenario of maximum call stack size overflow. So, in order our code to be optimized fast there are some that we need to improve in writing our code. So best practices are:

- Not change the shape of an object e.g. Let obj = {}; obj.name = "Michael"; is bad practice instead let obj = {name: "Michael";}

Now let's Processes and threads. Process are program holding a memory, inside the process there are threads that executes part of the process. E.g. now chrome runs multiple processes and this processes communicate through inter process communication(IPC). This gives more security. So the main process launches many process like

- Browser process
- Renderer process
- GPU process
- Network process …etc.

Let's handle the user input at first, so first the browser process parse the input. And there are two possibilities either the user will give URL input or search. After that the network process will receive command from browser process and sends HTTP request to the server. After that the network process waits and receives response from the server either download content type or redirect (if it's HTML, CSS and code status 200) to browser process and commands to prepare the renderer process. When the renderer process is gets ready the browser process start deleting the old data. When the rendering process confirm from browsing process it displays the response to the user and sends message to browser when finished.