What is the JavaScript engine?

A JavaScript engine is a software component that executes JavaScript code. The first JavaScript engines were mere interpreters, but all relevant modern engines use just-in-time compilation for improved performance. JavaScript engines are typically developed by web browser vendors, and every major browser has one.

ما هو محرك جافا سكريبت؟
محرك JavaScript هو أحد مكونات البرنامج الذي ينفذ تعليمات JavaScript البرمجية. كانت محركات JavaScript الأولى مجرد مترجمين فوريين، لكن جميع المحركات الحديثة ذات الصلة تستخدم الترجمة في الوقت المناسب لتحسين الأداء. عادةً ما يتم تطوير محركات JavaScript بواسطة بائعي متصفحات الويب، وكل متصفح رئيسي لديه واحد

What is the most popular JavaScript engine?

V8

The most popular JavaScript engine is V8. Designed for embedding in software, it powers Chrome, NodeJS, UXP, Deno and many other platforms. There are many other JS engines with different design goals. Some are designed for low-power or low-memory environments.

ما هو محرك جافا سكريبت الأكثر شعبية؟
V8
محرك جافا سكريبت الأكثر شعبية هو V8. تم تصميمه للتضمين في البرامج، وهو يعمل على تشغيل Chrome وNodeJS وUXP

و Deno والعديد من الأنظمة الأساسية الأخرى. هناك العديد من محركات JS الأخرى ذات أهداف تصميمية مختلفة. تم تصميم بعضها للبيئات منخفضة الطاقة أو الذاكرة المنخفضة.

History[edit]

The first JavaScript engine was created by Brendan Eich in 1995 for the Netscape Navigator web browser. It was a rudimentary interpreter for the nascent language Eich invented. (This evolved into the SpiderMonkey engine, still used by the Firefox browser.)

The first modern JavaScript engine was V8, created by Google for its Chrome browser. V8 debuted as part of Chrome in 2008, and its performance was much better than any prior engine.[2][3] The key innovation was just-in-time compilation, which can significantly improve execution times.

Other browser vendors needed to overhaul their interpreters to compete.[4] Apple developed the Nitro engine for its Safari browser, which had 30% better performance than its predecessor.[5] Mozilla leveraged portions of Nitro to improve its own SpiderMonkey engine.[6]

Since 2017, these engines have added support for WebAssembly. This enables the use of pre-compiled executables for performance-critical portions of page scripts.

تاريخ
تم إنشاء أول محرك JavaScript بواسطة Brendan Eich في عام 1995 لمتصفح الويب Netscape Navigator. لقد كان مترجمًا بدائيًا للغة الناشئة التي اخترعها إيتش. (تطور هذا إلى محرك SpiderMonkey، الذي لا يزال يستخدمه متصفح Firefox.)

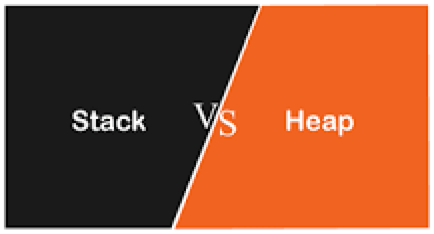
أول محرك جافا سكريبت حديث كان V8، الذي أنشأته جوجل لمتصفح كروم الخاص بها. ظهر محرك V8 لأول مرة كجزء من

متصفح Chrome في عام 2008، وكان أداؤه أفضل بكثير من أي محرك سابق.[2][3] وكان الابتكار الرئيسي هو التجميع في الوقت المناسب، والذي يمكن أن يحسن بشكل كبير أوقات التنفيذ.

كان بائعو المتصفحات الآخرون بحاجة إلى إصلاح مترجميهم الفوريين حتى يتمكنوا من المنافسة.[4] طورت شركة أبل محرك Nitro لمتصفحها Safari، والذي كان أداءه أفضل بنسبة 30% من سابقه.[5] استفادت موزيلا من أجزاء من Nitro لتحسين محرك SpiderMonkey الخاص بها.[6]

منذ عام 2017، أضافت هذه المحركات دعمًا لـ WebAssembly. يتيح ذلك استخدام الملفات التنفيذية المترجمة مسبقًا للأجزاء المهمة للأداء من البرامج النصية للصفحة

What is difference between stack and heap?



Stack provides static memory allocation, i.e., it is used to store the temporary variables. Heap provides dynamic memory allocation. By default, all the global variables are stored in the heap. It is a linear data structure means that elements are stored in the linear manner, i.e., one data after another

What is the difference between stack and heap function calls?

Data accessibility: Data in stack memory can only be accessed during an active function call, whereas data in heap memory remains accessible until it's manually deallocated or the program ends

Why stack is faster than heap?

In general, the stack is faster than the heap because memory is allocated and deallocated much more efficiently on the stack as we discussed earlier in the article but there's more to it than that. In Windows, the stack is typically implemented as a contiguous block of memory that is managed by the operating system

Is stack and heap in RAM?

Stack is used for static memory allocation and Heap for dynamic memory allocation, both stored in the computer's RAM . Variables allocated on the stack are stored directly to the memory and access to this memory is very fast, and it's allocation is dealt with when the program is compiled

Stack Memory

In programming, whenever a function is called, the program generates a new stack memory block for the function to utilize. This type of memory is located in the stack section of a program’s memory space, which is a reserved restricted memory area. This section is usually located at the top of the memory space and grows downward as more data is added.

Stack memory employs an automatic allocation and deallocation of memory that stores temporary data created by functions or procedures. Stack memory uses a “Last In, First Out” (LIFO) data structure, meaning that the most recently added item is the first to be removed. When the function or procedure is finished executing, the stack memory block is released automatically, and the program returns to the previous point of execution.

Stack memory is useful in managing memory usage as it avoids memory leaks due to its architecture. Moreover, the stack may overflow and crash in the case of many nested function calls.

Heap Memory

Compared to stack memory, heap memory operates dynamically, which basically means that the program can allocate and deallocate memory areas of different sizes when necessary. This allocation/deallocation of memory depends on the requirements that arise during runtime.

A program’s memory space consists of a heap section, which is reserved only for a program’s heap memory. The size of the heap partition is not fixed and can be dynamically adjusted at runtime.

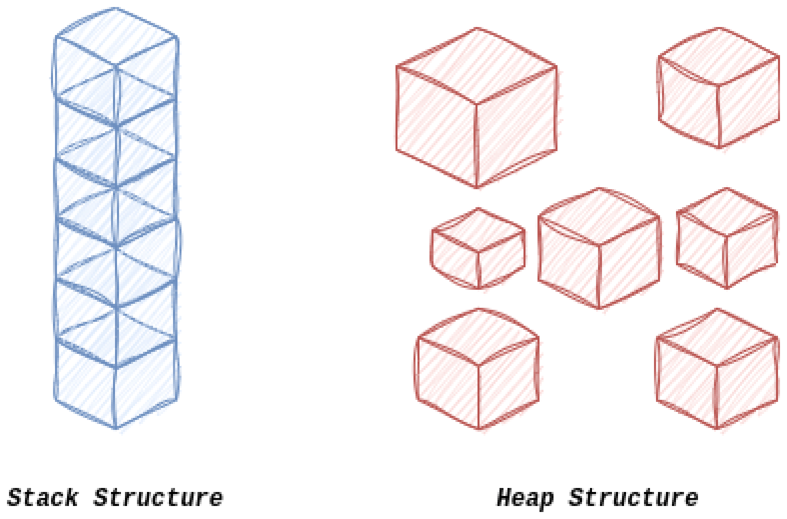
In addition, heap memory is not associated with a specific function or process. **The data in heap memory is not arranged in any specific pattern and can be reached in any order**. Throughout a program’s runtime, only useful areas of memory are retained in heap memory.

Consequently, heap memory is suitable for dealing with large, complex data structures such as tables, linked lists, and trees and facilitates memory sharing between different program parts.

Although, occasionally, heap memory may be more challenging and lead to memory leaks or other memory-related errors.

Stack vs. Heap Memory

Both stack and heap memory are located in a computer’s RAM (Random Access Memory):


Stack Structure
Heap Structure