Веб-сборка

WebAssembly — это новый тип кода, который можно запускать в современных веб-браузерах — это низкоуровневый язык, похожий на ассемблер, с компактным двоичным форматом, который работает почти с исходной производительностью и поддерживает такие языки, как C/C++, C# и Rust. с целью компиляции, чтобы они могли работать в Интернете. Он также предназначен для работы вместе с JavaScript, что позволяет им работать вместе.

В двух словах WebAssembly имеет огромное значение для веб-платформы — он позволяет запускать код, написанный на нескольких языках, в Интернете почти с естественной скоростью, а клиентские приложения работают в Интернете, что раньше было невозможно.

WebAssembly предназначен для дополнения и работы вместе с JavaScript — с помощью API-интерфейсов WebAssembly JavaScript вы можете загружать модули WebAssembly в приложение JavaScript и совместно использовать их функции. Это позволяет вам использовать преимущества производительности и мощности WebAssembly, а также выразительности и гибкости JavaScript в одних и тех же приложениях, даже если вы не знаете, как писать код WebAssembly.

И что еще лучше, он разрабатывается как веб-стандарт через рабочую группу W3C WebAssembly и группу сообщества при активном участии всех основных поставщиков браузеров.

Гиды Концепции WebAssembly Начните с прочтения основных концепций WebAssembly — что это такое, почему он так полезен, как он вписывается в веб-платформу (и не только) и как его использовать.

Компиляция нового модуля C/C++ в WebAssembly Когда вы написали код на C/C++, вы можете скомпилировать его в Wasm с помощью такого инструмента, как Emscripten . Давайте посмотрим, как это работает.

Компиляция существующего модуля C в WebAssembly Основной вариант использования WebAssembly — взять существующую экосистему библиотек C и позволить разработчикам использовать их в Интернете.

Компиляция из Rust в WebAssembly Если вы написали код на Rust, вы можете скомпилировать его в WebAssembly! В этом руководстве вы узнаете все, что вам нужно знать, чтобы скомпилировать проект Rust в Wasm и использовать его в существующем веб-приложении.

Загрузка и запуск кода WebAssembly После того, как у вас есть модуль Wasm, в этой статье рассказывается, как получить, скомпилировать и создать его экземпляр, объединив API JavaScript WebAssembly с API Fetch или XHR .

Использование JavaScript-API WebAssembly После того, как вы загрузили модуль Wasm, вы захотите его использовать. В этой статье мы покажем вам, как использовать WebAssembly через JavaScript API WebAssembly.

Экспортированные функции WebAssembly Экспортированные функции WebAssembly — это отражения функций WebAssembly в JavaScript, которые позволяют вызывать код WebAssembly из JavaScript. В этой статье описано, что они из себя представляют.

Понимание текстового формата WebAssembly В этой статье объясняется текстовый формат Wasm. Это низкоуровневое текстовое представление модуля Wasm, отображаемое в инструментах разработчика браузера при отладке.

Преобразование текстового формата WebAssembly в Wasm В этой статье представлено руководство о том, как преобразовать модуль WebAssembly, написанный в текстовом формате, в двоичный файл Wasm.

Справочник по API Справочник инструкций WebAssembly Справочная документация с интерактивными примерами для набора операторов WebAssembly.

JavaScript-интерфейс WebAssembly Этот объект действует как пространство имен для всех функций, связанных с WebAssembly.

WebAssembly.Global() Объект WebAssembly.Globalпредставляет собой экземпляр глобальной переменной, доступный как из JavaScript, так и импортируемый/экспортируемый в одном или нескольких WebAssembly.Moduleэкземплярах. Это позволяет динамически связывать несколько модулей.

WebAssembly.Module() Объект WebAssembly.Modulecодержит код WebAssembly без сохранения состояния, который уже скомпилирован браузером и может быть эффективно передан Workers и создан несколько раз.

WebAssembly.Instance() Объект WebAssembly.Instance— это исполняемый экземпляр объекта Module. Instanceобъекты содержат все экспортированные функции WebAssembly , которые позволяют вызывать код WebAssembly из JavaScript.

WebAssembly.compile() Функция WebAssembly.compile()компилирует двоичный код WebAssembly в WebAssembly.Moduleобъект.

WebAssembly.compileStreaming() Функция WebAssembly.compileStreaming()компилирует WebAssembly.Moduleнепосредственно из потокового базового источника.

WebAssembly.instantiate() Функция WebAssembly.instantiate()позволяет компилировать и создавать экземпляры кода WebAssembly.

WebAssembly.instantiateStreaming() Эта WebAssembly.instantiateStreaming()функция является основным API для компиляции и создания экземпляров кода WebAssembly, возвращая как a, Moduletак и его первый файл Instance.

WebAssembly.validate() Функция WebAssembly.validate()проверяет заданный типизированный массив двоичного кода WebAssembly.

WebAssembly.Memory() Объект WebAssembly.Memory— это изменяемый размер ArrayBuffer, который содержит необработанные байты памяти, к которым обращается объект Instance.

WebAssembly.Table() Объект WebAssembly.Tableпредставляет собой типизированный массив непрозрачных значений с изменяемым размером, например ссылки на функцию, к которым обращается объект Instance.

WebAssembly.Tag() Объект WebAssembly.Tagопределяет тип исключения WebAssembly, которое может быть выброшено в/из кода WebAssembly.

WebAssembly.Exception() Объект WebAssembly.Exceptionпредставляет собой исключение времени выполнения, выброшенное из WebAssembly в JavaScript или выброшенное из JavaScript в обработчик исключений WebAssembly.

WebAssembly.CompileError() Создает новый CompileErrorобъект WebAssembly.

WebAssembly.LinkError() Создает новый LinkErrorобъект WebAssembly.

WebAssembly.RuntimeError() Создает новый RuntimeErrorобъект WebAssembly.

Примеры ВАСМСобель См. наш репозиторий webassembly-examples для ряда других примеров. Технические характеристики Спецификация Интерфейс JavaScript WebAssembly # webassembly-namespace