WebAssembly入门

下一代web技术

什么是WebAssembly

WebAssembly 有一套完整的语义，实际上 **wasm** 是体积小且加载快的二进制格式， 其目标就是充分发挥硬件能力以达到原生执行效率

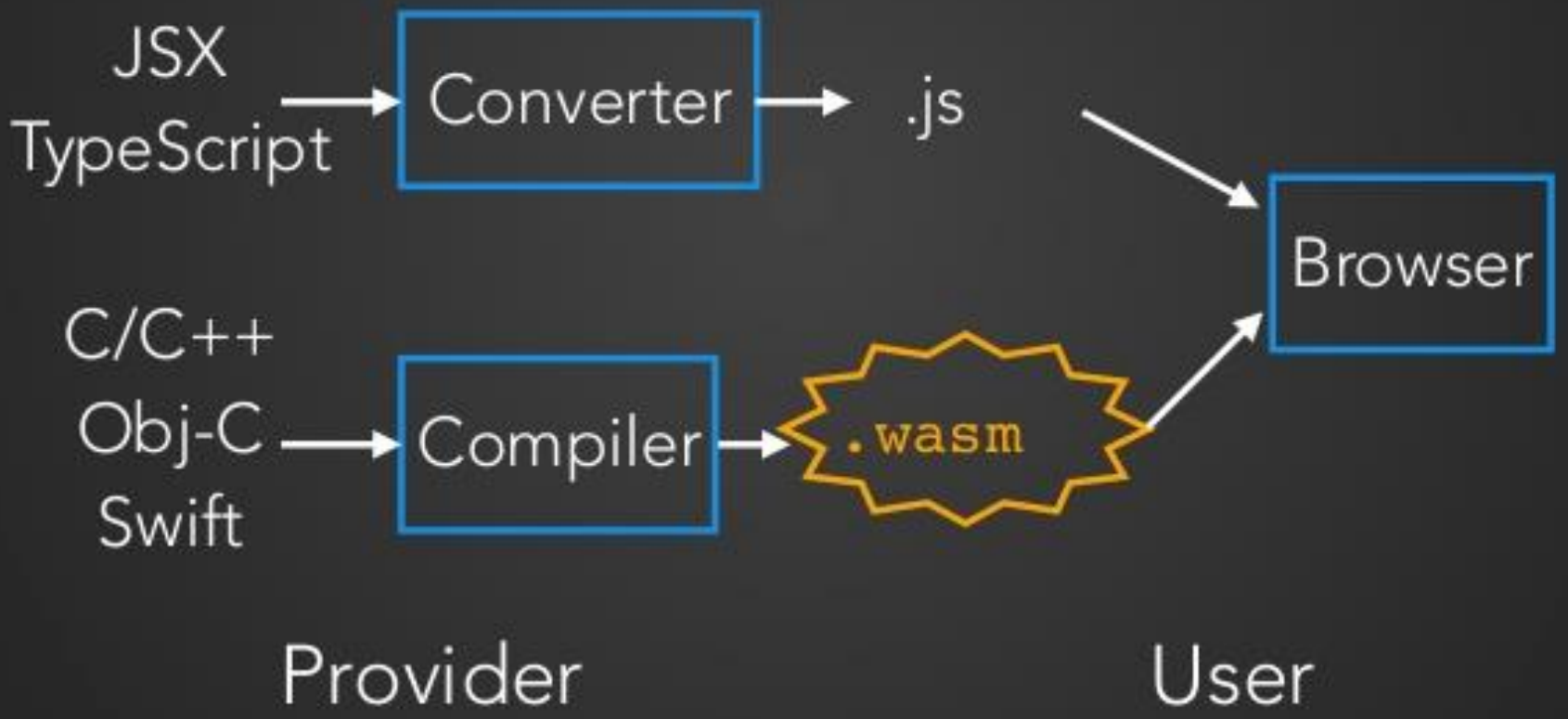
什么是WebAssembly

WebAssembly 是一种可以使用**非 JavaScript** 编程语言编写代码并且能在浏览器上运行的技术方案，实际上，是一种**新的字节码格式(当然还有很多优点)**

Web第四种语言

2019年12月5日，**WebAssembly** 正式成为 World Wide Web Consortium (W3C) 的标准，加入到了 **HTML、CSS 和 JavaScript** 的行列

WebAssembly



WebAssembly 的工作原理

WebAssembly 是除了 **JavaScript** 以外，另一种可以在网页中运行的编程语言。过去如果你想在浏览器中运行代码来对网页中各种元素进行控制，只有 JavaScript 这一种选择。

WebAssembly 与其他的汇编语言不一样，它不依赖于具体的物理机器。可以抽象地理解成它是**概念机器的机器语言**

WebAssembly 的优势

文件加载：WebAssembly 文件体积更小，所以下载速度更快
解析： 解码 WebAssembly 比解析 JavaScript 要快
编译和优化： 编译和优化所需的时间较少，因为在将文件推送到服务器之前已经进行了更多优化，JavaScript 需要为动态类型多次编译代码
重新优化： WebAssembly 代码不需要重新优化，因为编译器有足够的信息可以在第一次运行时获得正确的代码
执行： 执行可以更快，WebAssembly 指令更接近机器码
垃圾回收： 目前 WebAssembly 不直接支持垃圾回收，垃圾回收都是手动控制的，所以比自动垃圾回收效率更高
安全：可以放hash和签名等等

WebAssembly 的应用

WebAssembly 可用于视频和音频编解码器，图形和 3D，多媒体和游戏，密码计算或便携式语言实现等领域

代码演示，初步认识

```
fetch('./test.wasm').then(response =>
    response.arrayBuffer()
).then(bytes =>
    WebAssembly.compile(bytes)
).then(mod => {
    const instance = new WebAssembly.Instance(mod)
    const a = instance.exports

    console.log(a)
    console.time("测试 fib 速度: ");
    var re = a.fib(40);
    console.timeEnd("测试 fib 速度: ");

}
);
```