# Challenges in Impleneting client side compilation using WebAssembly

VJS PRANAVASRI, International Institute of Information Technology, India

WebAssembly compilers are tools that allow users to compile their code into the WebAssembly format. This format can be run on web browsers and other platforms that support the WebAssembly standard. We look at practical application of a webassembly compiler running on client side and study the challenges that arise in doing so. We present a prototype implementation of a webassembly compiler that runs on client side and discuss the challenges that we faced in implementing it. We also present a solution to the challenges that we faced.

CCS Concepts: • **Software and its engineering → Semantics**;

Additional Key Words and Phrases: WASM,Compilers,Software Engineering

**ACM Reference Format:**
VJS Pranavasri. 2018. Challenges in Impleneting client side compilation using WebAssembly. 1, 1 (November 2018), 9 pages. https://doi.org/XXXXXXX.XXXXXXX

## 1 INTRODUCTION

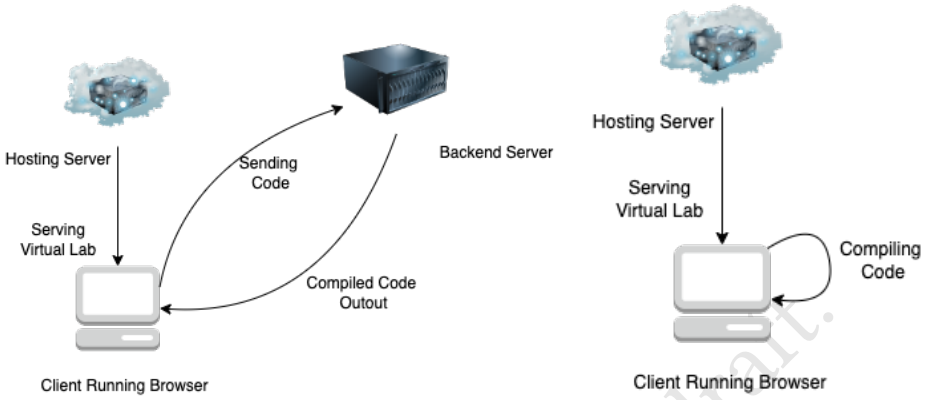
WebAssembly (Wasm)[5] is a web standard that defines a binary format and a corresponding assembly-like text format for executable code in web pages. The Wasm text format is designed as a portable, size-efficient, and embeddable format that runs with near-native performance and provides languages with low-level memory models such as C++ and Rust with a compilation target so that they can run on the web. A webassembly compiler is a tool that converts a code written in a high-level language like C++ or Rust into webassembly code. This code can be run on a web browser or on a web server.

The problem this paper tries to look at is being able to run codes natively on browser. WebAssembly is meant to be solving this problem, but it requires a compilation done in advance before actually hosting it.

I plan to research and find out ways in which a webassembly compiler will be able to compile High Level language codes to webassembly on the browser itself. This would allow for anyone to run(compile/interpret) any language natively on browser without having to have it compiled in advance. As part of the research I'll be looking at different possible ways to implement the compiler. The main focus of this paper would be a study on such web compilers which themselves are written in webassembly. We first start with looking at the available solutions for the same. We then go on with pyodide as a base for our implementation. We then discuss the challenges that we faced in implementing a webassembly compiler on client side. We will broadly try to validate the study on WebAssembly compiler bugs[10]

We will use Virtual Labs as a platform, where we test out our implementation for validaing our

Author's address: VJS Pranavasri, vjs.pranavasri@research.iiit.ac.in, International Institute of Information Technology, Professor CR Rao Road, Gachibowli, Hyderabad, Telangana, India, 500032.




(a) Server Side Compiling (b) Client Side Compiling

Fig. 1. Server Side Model vs Client Side Model



study.
We ask the following Research Questions to guide our study:

- **RQ1:** How will the implementation of a webassembly compiler help in the development of web applications?
- **RQ2:** What are the challenges in implementing a webassembly compiler on client side?
- **RQ3:** How big of an impact do bugs in WebAssembly compilers have on the implementation requirement of Virtual Labs?

## 2 VIRTUAL LABS

Virtual Labs is an initiative by the Ministry of Human Resource Development (MHRD), Government of India, under the aegis of the National Mission on Education through Information and Communication Technology (NMEICT). This is coordinated by 12 participating institutes of which IIIT Hyderabad is a part. The main goal is to make any lab or education available independently of the platform. Virtual Labs mainly help professors and students who do not have proper lab facilities available by virtually emulating the scenario.
One of the current challenges faced by Virtual Labs is the ability to run code. This is mainly because of the requirement of a backend server to run the code. As the number of students using Virtual Labs increases, the load on the backend server increases. This is mainly because of the fact that the code will be run on the server and the results are sent back to the client. This would take a toll on the resources of the server.
The solution to this problem is to run the code on the client side. This would eliminate the need for a backend server, and possibly also allow interacting with the lab without any network after the first time.
For the scope of this paper, we would be working on a Python based Virtual Lab and we will be using pyodide as a base for our study and implementation.

## 3 RELATED WORK

There is a lot of active study happening in the area of webassembly compilers, however we specifically focus on the ones that themseleves are written in webassembly that allows for client side compilations of other high/low level languages.

### WebAssembly

We take a lot of motivation from the paper that initially introduced webassembly to the world[5]. The paper explains the gap in web devlopment which cannot inherenyly be solved by using javascript. This in depth describes major components WebAssembly as a language and a tool. One of the main goal being able to run cross language codes.

### Study on WebAssembly

With the introduction of the new byte code format, a lot of analysis in terms of performance and security has been done. The paper [13] dicusses about the mechanisms of webassembly and how they verify the correctness of WASM. Then there ar multiple studies that go on to talk about performance of webassembly applications.
The paper [15] talks about how performance of web assembly based applications are and how different parameters like JIT(Just In Time) compilation and other factors affect the performance. Further studies focus on performance comparisions and energy consumption of webassembly applications compared to native applications[2]. An intersting study was one that used WebAssembly to study the memory model of javascript and propose changes to improve consistency[14].
A final research area is mainly geared towards security. The paper *Everything Old is New Again*[6] talks about binary security of web assembly, as we are again now moving towards assembly format for the web. We see a paper that talks about the security in place for web assembly compilers[12]. We would briefly refer to this to talk about security concerns that we find, and finally use the paper that does a performance analysis of webassembly compilers[9] as basis for our performance analysis.

### WebAssembly Applications & Compilers

We look at some existing applications and compiler implementations to gain a better understanding. One such is an implementation of the e-commerce website ebay using WASM[7]. Then we look at an implementation of tensorflow backend which has been ported to web through WASM[11], this allows for running models on web through client. Before heading further we look at one of the most popular compilers that is written for C, emscripten[16]. Emscripten is a compiler that compiles C/C++ to WASM. It is one of the most popular compilers for WASM.

### Bugs in WebAssembly Compilers

As we are focused more on the compilers that compile to WebAssembly, we use this paper as a base for our study[10]. This paper discusses the bugs that are present in the WebAssembly compilers. We use this to compare and study its effects the implementation of Virtual Labs.

We see that there isn't that major of work being done in fields of web assembly compilers, and nothing specific to running them on client side. This would vouch for the novelty of the paper and we use the remotely similar work as a base for our study. For this paper we will be specifically focussing on pyodide, a python distribution that runs on browser[1]. We will be using this as a base for our study and implementation. The source code for the same is available on github.[1]

[1]https://github.com/pyodide/pyodide

| Category | Number of Unique Issues |
|---|---|
| Asyncify Synchronous Code | 1 |
| Incompatible Data Type | 3 |
| Memory Model Differences | 7 |
| Other Infrastructure Bug | 12 |
| Emulating Native Environment | 1 |
| Supporting Web API's | 1 |
| Pyodide Specific | 21 |

Table 1. Categories of issues taken from [10]



## 4 METHODOLOGY

Initially we do a collection of bugs from Pyodide repo. We then go ahead to analyse these bugs and try to compare them to the bugs proosed by Romano et al. [10].

### 4.1 Bugs Collection

Our first goal is data collection. We follow the methods proposed followed by [10] and use Github Search API[3] and Github REST API[4] to collect all issues and pull requests. The project has a total of 880 Closed and 277 Open issues at the time of collection. For our use case, we filter out all the issues with label bug. This brings the total issues down tp 149 (closed) + 43 (open). We go through a detailed analysis of each issue and seggregate them based on the categories as shown in table 1. For the scope of the research course, we look at only the closed issues and filter out all duplicate bugs, won't-fix bugs and any bugs in the build time of pyodide. We majorly look at issues that are posed which are more focused with respect to pyodide as a framework itself or one of it's dependency CPython, emscripten or WebAssembly amongst others[2].

### 4.2 Analysis

From figure 2, Pyodide specific bugs are the highest with 21 of the bugs belonging to the category, With other Infrastructure bugs being the immediate next. We briefly describe the basis on which we have categorized the bugs.

#### Pyodide Specific

All the bugs that have occured purely due to the issue with coding styles, typos, or any other issue that is specific to pyodide. These bugs are mostly related to the core of the project and are not related to any of the dependencies. These also include the bugs that were introduced as a side effect of some other bug fix.

#### Other Infrastructure Bug

These bugs are related to the dependencies of pyodide. These include bugs related to CPython, emscripten, WebAssembly, Asyncify, etc. These bugs are mostly related to the way the dependencies are used in pyodide. These could be further divided based on a deeper study of all issues into other relavant categories. But at this time we broadly separate all external dependent bugs into this category.

[2]The complete list of bugs is avaialble on sheets https://stagbin.tk/pyodide_issues

### Memory Model Differences

These are bugs which look at differences in the data types or the way storage is allocated to variables. For example transferring byte objects between pyodide and javascript[3]. These bugs are mostly related to the way the data is stored in the memory and how it is accessed.

### Incompatible Data Type

These include issues that arise due to incompatibility of data types when compiling from one language to another.

### Supporting Web API's

These are bugs which look at the support for web API's. Web Api's are the set of functions that are available in the browser. These are mostly related to the way the browser interacts with the code. For example, the issue of `Promise` object not being available in Pyodide.

### Emulating Native Environment

These are bugs that depend on native environment such as POSIX threads or POSIX signals.

### Asyncify Synchronous Code

These are bugs that arise due to the use of synchronous code in the browser.

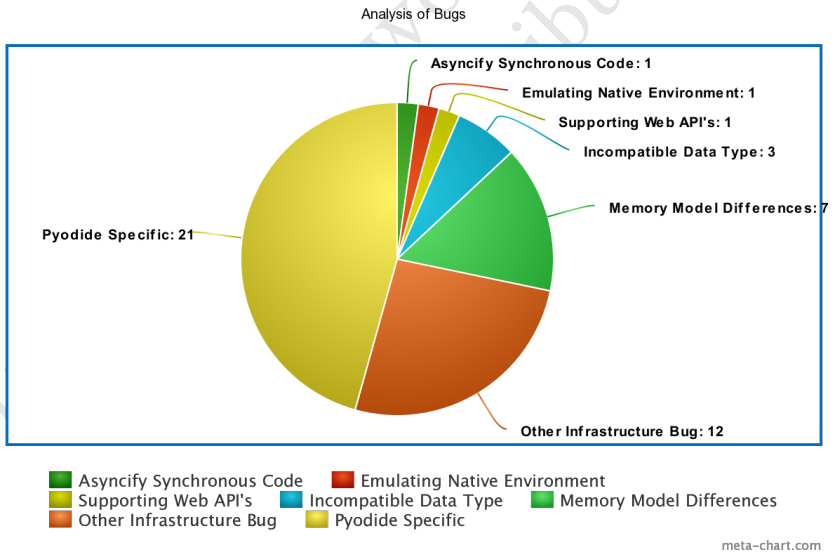



Fig. 2. Analysis of bugs

## 4.3 Impact of Bugs

The impact of bugs in a bigger scope varies from project to project. For instance a bug that exists when installing scipy[4] in pyodide would be more relavant to a project focused towards Machine Learning. But for our use case with virtual labs, we are looking at the bugs that interfere with the

[3]This issue can be found in #749 in pyodide issues[8]
[4]https://github.com/pyodide/pyodide/pull/2348

requirements of Client Side Virtual Labs implementation.
From the study, we see majority of the bugs are geared towards issues specific to pyodide, installing libraries and to memory management, most of which do not directly impact the implementation of virtual labs. But there are a few bugs that are more relavant to our use case. To be specific we look at a major bugs we have identified.

### Interrupting the execution of code

Due to single threaded nature of javascript and the lack of signal handling, it is not possible to interrupt the execution of code. This is a major issue when it comes to implementing python virtual labs. For instance, in a virtual lab, the user is expected to be able to interrupt the execution of code at any point of time or to automatically stop the execution in case of an infinite loop. Pyodide suggests a solution with use of web workers[5]. But this depends on Browsers having support for SharedArrayBuffer[6]. This is not supported by all browsers and is still in experimental phase. To enable this, the server hosting the virtual lab needs to have the correct headers set.

## 5 IMPLEMENTATION

The main goal for realising client side compilation is to extend the existing virtual labs to add support to work as an online coding platform. The current implementation which can be used to compile and run code written in different languages, starting with python using pyodide.

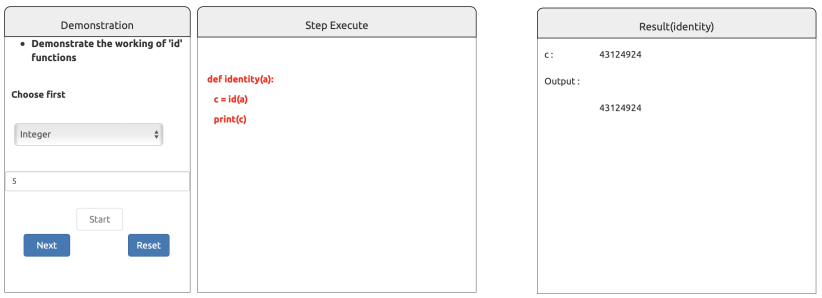



Fig. 3. The current implementation of virtual labs

For implementation, we closesly follow the documentation of Pyodide, and implement a very lightweight version of it. The screenshot of the same can be found in Fig 4[7]. Considering all the requirements, we have a very basic implementation that takes care of:

- Loading the python interpreter and running the code client side.
- Allowing a user to interrupt the execution of the code.
- Automatically stopping the execution of the code after a certain time limit.
- Disabling editing to a specific section of the code.

Intially we directly implemented using JS, and then from the bug analysis and requirement study we realised the requirement of code interrupt as discussed in 4.3. We go ahead with the implementation of the same using web workers. The interruption is especially the focus as it

[5]https://pyodide.org/en/stable/usage/webworker.html
[6]https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/SharedArrayBuffer
[7]Source code for the same can be found at https://github.com/vjspranav/Pyodide-implementation

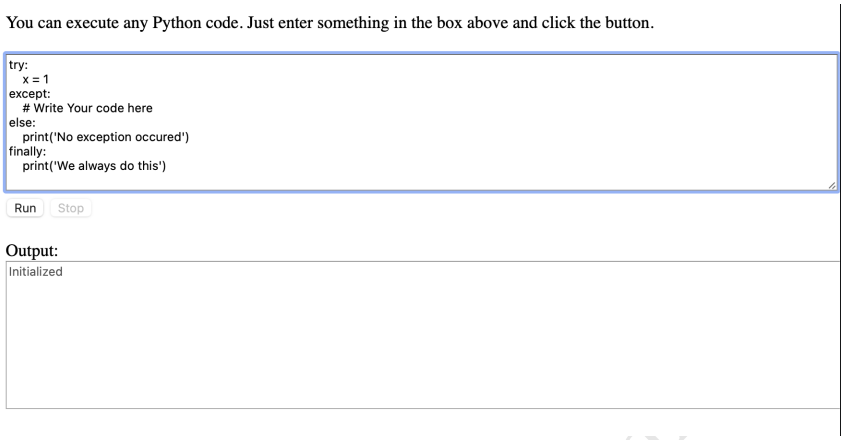



Fig. 4. Client side compiling using Pyodide

can heavily influence the user's experience and can cause severe lags on the browser in cases of infinitiley running codes.

## 6 DISCUSSION

Multiple online coding and education platforms like Virtual Labs, would need an exclusive backend server running if they intend on having code compilation. WebAssembly has brought in many new innovations, and if we could utilize it's power to allow the compilation on browser itself it would save multiple such organisations a huge cost of having a backend server also at the same time, sideloading all the work to a client's browser. We discuss how we come to a solution for each of the Research Questions that we had asked.

### 6.1 RQ1: How will the implementation of a webassembly compiler help in the development of web applications?

We do not have a strong ground to state a fixed answer here, but a very obvious positive impact would be the reduction in the cost of hosting a web application. The cost of hosting a web application is directly proportional to the number of users it has. If we could reduce the cost of hosting a web application, it would be a huge benefit for the organisation.[8]
This would also give an organisation power to use service worker and load these compilers without internet connection. This would allow the users to use the labs even when they are offline.
We discuss how we could add more validation to this in future work.

### 6.2 RQ2: What are the challenges in implementing a webassembly compiler on client side?

The only challenge that we face was the fact that multiple features such as threads, SIMD, and GC are not supported by browsers. This is a major challenge as these features are very important for a compiler to be able to compile a high level language.
Majority of these do not have any impact on our specific requirement which is geared towards a basic Python Programming Lab.

[8]When we talk about cost of hosting a web application, we specifically are talking about the cost of hosting a backend server for compiling codes.

### 6.3 RQ3: How big of an impact do bugs in WebAssembly compilers have on the implementation requirement of Virtual Labs?

Although there are many bugs that exist in pyodide both resolved and unresolved, we see that the impact of these bugs is not very big on our implementation. We have been able to implement a basic Python Programming Lab with the help of pyodide.
The only issue that we had was to setup interrupts due to lack of support for signals webassembly. This as discussed was resolved using Web Workers which are not supported by all browsers widely.

## 7 CONCLUSION

In this paper we try to show the importance of having client side compilation. We conduct our study using Pyodide as a base, and answer the questions pertaining to bugs and implementation in a project. We realize the possibilities this brings to an organisation both interms of performarnce and resources required.
We also perform a bug analysis using a previously done research and validate the approaches and bugs from that paper.
We believe there is a huge scope by moving the compilation to the client side for all services that require a backend server for processing. This is because, even mobile devices are releasing with processors sufficient enough to run even AI models. This would greatly improve the lags caused by network latency and server loads and greatly improve the user experience.

## 8 FUTURE WORK

As the scope of this research was small we have a lot of suggestions for future work. We could see a lot of bugs that could be further categorised to be more specific to pyodide, doing a continuation of the comparitive study would help better in validating the bugs paper.
We could also look at the performance of the pyodide compiler and see how it compares to an actual backend compiler, this would ideally be done through A/B testing on the virtual labs platform itself.
A study on user experience and user feedback could be done to validate the proposal.
The proposal could be validated by a larger audience with a working solution deployed on the virtual labs platform. Considering the small time frame it was not possible to get the solution deployed on the platform.



## REFERENCES

[1] Pyodide contributors. 2021. Version 0.21.3. https://pyodide.org/en/stable/
[2] João De Macedo, Rui Abreu, Rui Pereira, and João Saraiva. 2022. WebAssembly versus JavaScript: Energy and Runtime Performance. In *2022 International Conference on ICT for Sustainability (ICT4S)*. 24–34. https://doi.org/10.1109/ICT4S55073.2022.00014
[3] Search Github Docs. 2020. Github search api. https://docs.github.com/en/rest/reference/search
[4] Github. 2019. Github REST API. https://developer.github.com/v3/
[5] Andreas Haas, Andreas Rossberg, Derek L. Schuff, Ben L. Titzer, Michael Holman, Dan Gohman, Luke Wagner, Alon Zakai, and JF Bastien. 2017. Bringing the Web up to Speed with WebAssembly. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation* (Barcelona, Spain) *(PLDI 2017)*. Association for Computing Machinery, New York, NY, USA, 185–200. https://doi.org/10.1145/3062341.3062363
[6] Daniel Lehmann, Johannes Kinder, and Michael Pradel. 2020. Everything Old is New Again: Binary Security of WebAssembly. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 217–234. https://www.usenix.org/conference/usenixsecurity20/presentation/lehmann
[7] Senthil Padmanabhan and Pranav Jha. 2020. Webassembly at ebay: A real-world use case.
[8] Pyodide. 2021. Pyodide issues. https://github.com/pyodide/pyodide/issues/

[9] Micha Reiser and Luc Bläser. 2017. Accelerate JavaScript applications by cross-compiling to WebAssembly. 10–17. https://doi.org/10.1145/3141871.3141873

[10] Alan Romano, Xinyue Liu, Yonghwi Kwon, and Weihang Wang. 2021. An Empirical Study of Bugs in WebAssembly Compilers. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, Piscataway, NJ, USA, 42–54. https://doi.org/10.1109/ASE51524.2021.9678776

[11] Daniel Smilkov, Nikhil Thorat, and Ann Yuan. 2020. Introducing the WebAssembly backend for TensorFlow. js. *Retrieved February* 3 (2020), 2022.

[12] Quentin Stiévenart, Coen De Roover, and Mohammad Ghafari. 2021. The Security Risk of Lacking Compiler Protection in WebAssembly. In *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*. 132–139. https://doi.org/10.1109/QRS54544.2021.00024

[13] Conrad Watt. 2018. Mechanising and Verifying the WebAssembly Specification. In *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs* (Los Angeles, CA, USA) *(CPP 2018)*. Association for Computing Machinery, New York, NY, USA, 53–65. https://doi.org/10.1145/3167082

[14] Conrad Watt, Andreas Rossberg, and Jean Pichon-Pharabod. 2019. Weakening WebAssembly. *Proc. ACM Program. Lang.* 3, OOPSLA, Article 133 (oct 2019), 28 pages. https://doi.org/10.1145/3360559

[15] Yutian Yan, Tengfei Tu, Lijian Zhao, Yuchen Zhou, and Weihang Wang. 2021. Understanding the Performance of Webassembly Applications. In *Proceedings of the 21st ACM Internet Measurement Conference* (Virtual Event) *(IMC '21)*. Association for Computing Machinery, New York, NY, USA, 533–549. https://doi.org/10.1145/3487552.3487827

[16] Alon Zakai. 2011. Emscripten: an LLVM-to-JavaScript compiler. In *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*. 301–312.