# Servo: Prototype ways of splitting up the script crate

## Personal Details


- Name: Peter Hrvola
- Email: peter.hrvola@hotmail.com
- IRC nick: retep007
- Telephone: +420606976702
- Other contact methods:
- Country of residence: Slovakia
- Primary language: Slovak
- Timezone: GMT+1


## Abstract


The aim of this project is to investigate and propose a plan for separation of Servo script crate. Script crate lays in the core of the Servo and requires a large amount of memory to build.


## Project Proposal

I'm applying for: "Servo: Prototype ways of splitting up the script crate". Since 2014 the script crate has been growing in size, making it slow to build. In December 2017, new milestone has been met. Development build has exceed 4GB memory limit, making it impossible to build on 32-bit computers.
This crate consist from a large amount of tightly coupled hand-written and generated code. My goal is to propose a solution of splitting this crate to at least 2 crates.
The problem of separation consists in WebIDLs used to describe the interfaces, data types and their properties. During compilation WebIDLs are used to generate rust "bindings" using python script. Bindings are later used in hand written code in `/dom` directory. Compilation of such large crate is expensive on CPU and RAM.
Smaller crates have multiple advantages. It allows quicker development, rebuilding only small partition of changed code. WebIDLs are changed less often than hand-written code, which makes it a perfect candidate for separation. Splitting to smaller crates lowers maximal memory usage during compilation of this crate.
I've solved provided GSOC splitting challenge as an exercise for splitting crates. Also I created a fork of the Servo repo which does not use code generation but code is committed for better workflow. I have already minimized number of files in script crate in this fork. Basically it has unmodified `dom/Attr.rs` and everything else was modified to successfully compile. The fork can be later used for experimentation and ideas can be tested easier by expanding number of files in this fork. I created the fork in order to get experience with dependencies inside crate and prepare playground for the project.

## Proposed Ideas

I have created these ideas based on already mentioned ideas in https://github.com/servo/servo/issues/1799 which however might be a bit outdated, and problem has become larger ever since. These are only ideas to consider trying out.

Jdm idea: Single trait with associated types for every WebIDL interface, and every generic method gets parameterized over the trait and uses T::Interface rather than Interface.

Asajeffrey idea: Make each FooMethods trait parametric, rather than each method inside FooMethods. Something like:

```
trait DOMTypes {
  type Foo: FooMethods<Self>;
  ...
}
trait FooMethods<DOM: DOMTypes> {
  fn bar(x: DOM::Bar) -> DOM::Baz;
  ...
}
struct DOM;
impl DOMTypes for DOM {
  type Foo = Foo;
  ...
}

struct Foo { ... }
impl FooMethods<DOM> for Foo { ... }
```

Mine idea:
- Create a new crate script_binding which uses `script_traits`
- Traits for all `dom::*` struct would be added to `script_traits`
- Script crate will implement `script_traits`

## Schedule of Deliverables

I'm planning to devote 5 hours each day. If behind schedule I'm willing to devote more. During GSOC I have no other commitments. However I will require 3 free days for the final bachelor examination.

The final deliverables consist of:

- Documentation of all methods of separating crate that have been tested, encountered problems and outcomes
- Pros and cons for all of the methods for separation

- Plan for splitting `script` crate

Staring on April 20, when Google announces the accepted students. My planned schedule is:

- Week 1: Understand complex dependencies in crate; discuss on why things are organized in a way they are
- Week 2 - Week 3: Get to know the community; ask for ideas about separation of crate and evaluate ideas
- Week 4 - Week 8: Implementing best proposed splitting idea
- Week 9 - Week 11: Implementing 2nd best proposed splitting idea
- Week 12 - Week 14: Implementing 3rd best proposed splitting idea
- Week 15 - Week 16: Finalizing documentation; future plans

Note these are approximated schedules. Please allow adjustments according to expected outcome, its progress and benefits. If viable solution of splitting crate is found and approved by community, I would start separating the crate, but leave last 2 weeks for documentation and finishing.

I have scheduled ~4 weeks for each idea. This time is large enough to find all problems that would arise during implementation and also it is enough time for experimenting with representative sample of code. If promising solution is found this time span can be increased.

## Open Source Development Experience

https://github.com/rust-lang/rust/commits?author=retep007 - fixes in rust

https://github.com/retep007/apache_log_security - bachelor thesis in rust (WIP)

https://github.com/retep007/node-big-xml/ - added option for bzip stream to fork of unmaintained package

https://github.com/diesel-rs/diesel reviewer (I focus on documentation, which is not very understandable)

https://github.com/retep007/security-log access log scanner (Haskell)

## Work/Internship Experience

**Sygic Travel** - *Frontend developer (part-time)*

2015 - 2017

**Sygic Travel** - *Backend developer (part-time)*

2017 - 2018

## Academic Experience

**Masaryk University** - *Computer systems (bachelor)*

2015 - 2018

**Masaryk University** – assistant seminar tutor for “Principles of low level programming”
2018 Spring

I attend cybersecurity lab from which I’ve got an opportunity to attend at a team competition Locked shields 2017 (cybersecurity for NATO government agencies) and we ended up first.

## Why Me

Job of separating `script` crate is important and over time it will become only harder to accomplish. I strongly believe I have the programming experience necessary to complete this project successfully.

At the moment I’m getting increasingly involved in contributing to rust compiler, which is an awesome source to learn both development on large open source project and rust itself. I’m also reviewer in diesel-rs, because I have had experience with ORMs in other languages from what I learned how a good ORM can speed up development and different approaches in creating ORMs.

## Why Mozilla

I strongly believe in its mission and I would be really proud of helping to create the open source browser and wake up every morning with feeling that millions of people are using your code.