




# Rust WebAssembly in Electron

Daniel Maslowski



# Agenda

-  Rust and WebAssembly
-  Rust Wasm in Fiedka
-  Rust Wasm in dtvis

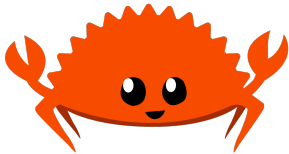
# Rust and WebAssembly

What if...



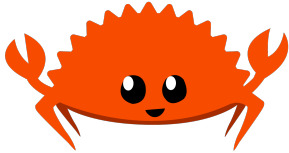
# What if...

... we compile Rust...



# What if...

... we compile Rust...

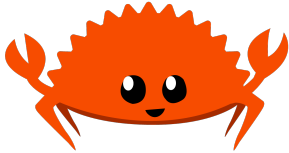


... to Wasm...



# What if...

... we compile Rust...



... to Wasm...

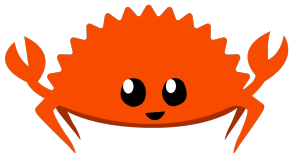


... and use it in an app?



# What if...

... we compile Rust...



... to Wasm...



... and use it in an app?



Magic happens - we can use native code on web platforms!



# Howto

# Howto

Getting started

<https://lannonbr.com/blog/2020-01-07-rust-wasmpack/>

<https://rustwasm.github.io/docs/wasm-pack/>



# Howto

Getting started

<https://lannonbr.com/blog/2020-01-07-rust-wasmpack/>

<https://rustwasm.github.io/docs/wasm-pack/>

TL;DR

```
cargo install wasm-pack
```

```
wasm-pack new my-rust-wasm-foo
```



# Howto

## Getting started

<https://lannonbr.com/blog/2020-01-07-rust-wasmpack/>

<https://rustwasm.github.io/docs/wasm-pack/>

## TL;DR

```
cargo install wasm-pack
```

```
wasm-pack new my-rust-wasm-foo
```

## The glue

<https://github.com/wasm-tool/wasm-pack-plugin>

<https://rustwasm.github.io/docs/wasm-pack/tutorials/hybrid-applications-with-webpack/using-your-library.html>



# Howto

## Getting started

<https://lannonbr.com/blog/2020-01-07-rust-wasmpack/>

<https://rustwasm.github.io/docs/wasm-pack/>

## TL;DR

```
cargo install wasm-pack
```

```
wasm-pack new my-rust-wasm-foo
```

## The glue

<https://github.com/wasm-tool/wasm-pack-plugin>

<https://rustwasm.github.io/docs/wasm-pack/tutorials/hybrid-applications-with-webpack/using-your-library.html>

## More glue

```
cargo add gloo-utils
```



# The Rust side



# The Rust side

```
extern crate wasm_bindgen;
use gloo_utils::format::JsValueSerdeExt;
use serde::{Deserialize, Serialize};
use wasm_bindgen::prelude::*;
/// ...
#[derive(Serialize, Deserialize)]
struct Foo {
    bar: u32,
    baz: String,
}
#[wasm_bindgen]
pub fn some_fun(data: JsValue) -> JsValue {
    /// ...
    let foo = Foo::new { bar: 42, baz: "Rust Wasm" };
    JsValue::from_serde(&foo).unwrap()
}
```



# The JavaScript side

```
import { some_fun } from "./rs/pkg";  
  
/* ... */  
    const res = some_fun({ woopWoop: 1337 });  
    console.info(res);  
/* ... */
```



# The JavaScript side

```
import { some_fun } from "./rs/pkg";  
  
/* ... */  
    const res = some_fun({ woopWoop: 1337 });  
    console.info(res);  
/* ... */
```

But that is synchronous and blocking!

# The JavaScript side

```
import { some_fun } from "./rs/pkg";  
  
/* ... */  
    const res = some_fun({ woopWoop: 1337 });  
    console.info(res);  
/* ... */
```

But that is synchronous and blocking!

<https://rustwasm.github.io/wasm-bindgen/reference/js-promises-and-rust-futures.html>

[https://rustwasm.github.io/wasm-bindgen/api/wasm\\_bindgen\\_futures/](https://rustwasm.github.io/wasm-bindgen/api/wasm_bindgen_futures/)



# Rust Wasm in Fiedka



# What is Fiedka again?



# What is Fiedka again?

Fiedka is the graphical desktop  
**f**irmware analyzer and **e**ditor.



<https://fiedka.app>

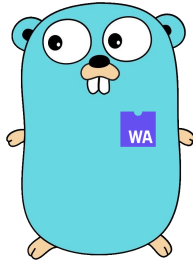
# What is Fiedka again?

Fiedka is the graphical desktop **f**irmware analyzer and **e**ditor.



<https://fiedka.app>

The current backend is written in Go and runs in Web Assembly.



# What is Fiedka again?

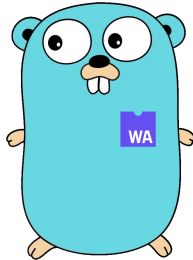
Fiedka is the graphical desktop **f**irmware analyzer and **e**ditor.



<https://fiedka.app>

Rust support is being added because there are tools written in it, e.g.,  
Romulan: <https://github.com/system76/romulan>

The current backend is written in Go and runs in Web Assembly.



## DEMO: Rust Wasm in Fiedka



# Rust Wasm in dtvis

# What is dtvis?

---

<sup>1</sup><https://devicetree.org>



# What is dtvis?

*dtvis* is a new project - a DeviceTree<sup>1</sup> visualizer.

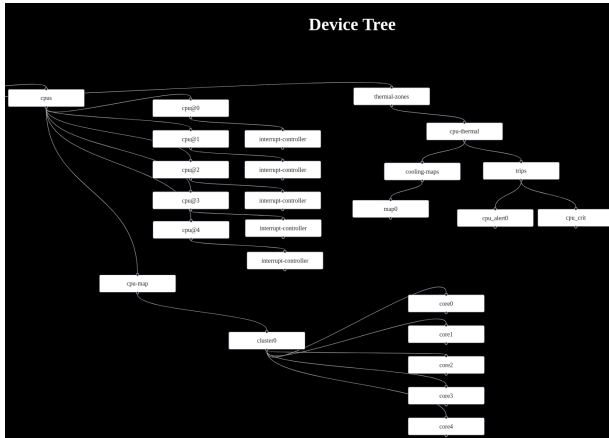
---

<sup>1</sup><https://devicetree.org>



# What is dtvis?

*dtvis* is a new project - a DeviceTree<sup>1</sup> visualizer.



<https://github.com/platform-system-interface/dtvis>

<sup>1</sup><https://devicetree.org>

# ... so what is a Device Tree?



IEEE 1275<sup>2</sup> Standard for Boot (Initialization Configuration)  
Firmware: Core Requirements and Practices / Open Firmware<sup>3</sup>



---

<sup>2</sup><https://standards.ieee.org/ieee/1275/1932/>

<sup>3</sup><https://www.openfirmware.info/data/docs/of1275.pdf>



## ... so what is a Device Tree?




-  IEEE 1275<sup>2</sup> Standard for Boot (Initialization Configuration) Firmware: Core Requirements and Practices / Open Firmware<sup>3</sup>
-  describing hardware topology for non-discoverable devices

---

<sup>2</sup><https://standards.ieee.org/ieee/1275/1932/>

<sup>3</sup><https://www.openfirmware.info/data/docs/of1275.pdf>

## ... so what is a Device Tree?





-  IEEE 1275<sup>2</sup> Standard for Boot (Initialization Configuration) Firmware: Core Requirements and Practices / Open Firmware<sup>3</sup>
-  describing hardware topology for non-discoverable devices
-  Linux, U-Boot, Zephyr, OLPC, FreeBSD and other projects use it
  - ▶ including Apple back in the days

---

<sup>2</sup><https://standards.ieee.org/ieee/1275/1932/>

<sup>3</sup><https://www.openfirmware.info/data/docs/of1275.pdf>

## ... so what is a Device Tree?

-  IEEE 1275<sup>2</sup> Standard for Boot (Initialization Configuration) Firmware: Core Requirements and Practices / Open Firmware<sup>3</sup>
-  describing hardware topology for non-discoverable devices
-  Linux, U-Boot, Zephyr, OLPC, FreeBSD and other projects use it
  - ▶ including Apple back in the days
-  not too elegant, attached to kernel via bindings

---

<sup>2</sup><https://standards.ieee.org/ieee/1275/1932/>

<sup>3</sup><https://www.openfirmware.info/data/docs/of1275.pdf>



## ... so what is a Device Tree?

- 👤 IEEE 1275<sup>2</sup> Standard for Boot (Initialization Configuration) Firmware: Core Requirements and Practices / Open Firmware<sup>3</sup>
- 👤 describing hardware topology for non-discoverable devices
- 👤 Linux, U-Boot, Zephyr, OLPC, FreeBSD and other projects use it
  - ▶ including Apple back in the days
- 👤 not too elegant, attached to kernel via bindings
- 👤 only few specified fields, most are “as someone wrote them”
  - ▶ compare e.g. Amlogic vs Allwinner SoC based trees

---

<sup>2</sup><https://standards.ieee.org/ieee/1275/1932/>

<sup>3</sup><https://www.openfirmware.info/data/docs/of1275.pdf>

## ... so what is a Device Tree?

- 👤 IEEE 1275<sup>2</sup> Standard for Boot (Initialization Configuration) Firmware: Core Requirements and Practices / Open Firmware<sup>3</sup>
- 👤 describing hardware topology for non-discoverable devices
- 👤 Linux, U-Boot, Zephyr, OLPC, FreeBSD and other projects use it
  - ▶ including Apple back in the days
- 👤 not too elegant, attached to kernel via bindings
- 👤 only few specified fields, most are “as someone wrote them”
  - ▶ compare e.g. Amlogic vs Allwinner SoC based trees
- 👤 can range from a few hundred to a thousand nodes

---

<sup>2</sup><https://standards.ieee.org/ieee/1275/1932/>

<sup>3</sup><https://www.openfirmware.info/data/docs/of1275.pdf>

## ... so what is a Device Tree?

- 👤 IEEE 1275<sup>2</sup> Standard for Boot (Initialization Configuration) Firmware: Core Requirements and Practices / Open Firmware<sup>3</sup>
- 👤 describing hardware topology for non-discoverable devices
- 👤 Linux, U-Boot, Zephyr, OLPC, FreeBSD and other projects use it
  - ▶ including Apple back in the days
- 👤 not too elegant, attached to kernel via bindings
- 👤 only few specified fields, most are “as someone wrote them”
  - ▶ compare e.g. Amlogic vs Allwinner SoC based trees
- 👤 can range from a few hundred to a thousand nodes
- 👤 the tree is a lie; there are cycles, e.g., power supplies and clocks

---

<sup>2</sup><https://standards.ieee.org/ieee/1275/1932/>

<sup>3</sup><https://www.openfirmware.info/data/docs/of1275.pdf>

## Previous attempts

There were discussions on tooling at Linux Plumbers<sup>4</sup>, partially stalled.

---

<sup>4</sup>[https://elinux.org/images/8/83/Plumbers\\_2016\\_dt\\_device\\_tree\\_tools.pdf](https://elinux.org/images/8/83/Plumbers_2016_dt_device_tree_tools.pdf)



# Previous attempts

There were discussions on tooling at Linux Plumbers<sup>4</sup>, partially stalled.



Component Inspector (by Freescale, now NXP)



- proprietary, closed source Eclipse plugin



- was part of QorIQ Configuration Suite, no longer available

---

<sup>4</sup>[https://elinux.org/images/8/83/Plumbers\\_2016\\_dt\\_device\\_tree\\_tools.pdf](https://elinux.org/images/8/83/Plumbers_2016_dt_device_tree_tools.pdf)



# Previous attempts

There were discussions on tooling at Linux Plumbers<sup>4</sup>, partially stalled.



Component Inspector (by Freescale, now NXP)

- ▶ proprietary, closed source Eclipse plugin
- ▶ was part of QorIQ Configuration Suite, no longer available



<https://github.com/dev-0x7C6/fdt-viewer>

- ▶ mixed tree + hex/text viewer, C++ + Qt
- ▶ supports dtb, dtbo (overlay) and itb (FIT image)

---

<sup>4</sup>[https://elinux.org/images/8/83/Plumbers\\_2016\\_dt\\_device\\_tree\\_tools.pdf](https://elinux.org/images/8/83/Plumbers_2016_dt_device_tree_tools.pdf)

<sup>5</sup><https://www.spinics.net/lists/devicetree-spec/msg00950.html>

# Previous attempts

There were discussions on tooling at Linux Plumbers<sup>4</sup>, partially stalled.



Component Inspector (by Freescale, now NXP)

- ▶ proprietary, closed source Eclipse plugin
- ▶ was part of QorIQ Configuration Suite, no longer available



<https://github.com/dev-0x7C6/fdt-viewer>

- ▶ mixed tree + hex/text viewer, C++ + Qt
- ▶ supports dtb, dtbo (overlay) and itb (FIT image)



<https://github.com/bmx666/dtv-demo>

- ▶ “RFC - DTV (Device Tree Visualiser)” on mailing list<sup>5</sup>
- ▶ dt\_s\_only, more of a text editor, Python + Qt6

---

<sup>4</sup>[https://elinux.org/images/8/83/Plumbers\\_2016\\_dt\\_device\\_tree\\_tools.pdf](https://elinux.org/images/8/83/Plumbers_2016_dt_device_tree_tools.pdf)

<sup>5</sup><https://www.spinics.net/lists/devicetree-spec/msg00950.html>

<sup>6</sup><https://marketplace.visualstudio.com/items?itemName=ploreface.devicetree>

# Previous attempts

There were discussions on tooling at Linux Plumbers<sup>4</sup>, partially stalled.



Component Inspector (by Freescale, now NXP)

- ▶ proprietary, closed source Eclipse plugin
- ▶ was part of QorIQ Configuration Suite, no longer available



<https://github.com/dev-0x7C6/fdt-viewer>

- ▶ mixed tree + hex/text viewer, C++ + Qt
- ▶ supports dtb, dtbo (overlay) and itb (FIT image)



<https://github.com/bmx666/dtv-demo>

- ▶ “RFC - DTV (Device Tree Visualiser)” on mailing list<sup>5</sup>
- ▶ dt\_s\_ only, more of a text editor, Python + Qt6



VS Code plugin `plorefice.devicetree`<sup>6</sup>

- ▶ syntax highlighting + collapsing
- ▶ could be enhanced with `dtvis` :-)

---

<sup>4</sup>[https://elinux.org/images/8/83/Plumbers\\_2016\\_dt\\_device\\_tree\\_tools.pdf](https://elinux.org/images/8/83/Plumbers_2016_dt_device_tree_tools.pdf)

<sup>5</sup><https://www.spinics.net/lists/devicetree-spec/msg00950.html>

<sup>6</sup><https://marketplace.visualstudio.com/items?itemName=plorefice.devicetree>



## DEMO: Rust Wasm in dtvis

# Related

Fiedka the Firmware Editor (OSFC 2021)

<https://www.osfc.io/2021/talks/fiedka-the-firmware-editor/>

Platform System Interface - Design und Evaluation holistischer  
Computerarchitektur (rC3 2022)

<https://media.ccc.de/v/fire-shonks-2022-49154-platform-system-interface-design-und-evaluation-holistischer-computerarchitektur>

Hack the Gadget! (MRMCD 2023)

<https://talks.mrmcd.net/2023/talk/SLLVT8/>



Thank you! :)



# Follow Me



Daniel Maslowski

<https://github.com/orangecms>  
<https://twitter.com/orangecms>  
<https://mastodon.social/@cyrevolt>  
<https://youtube.com/@cyrevolt>  
<https://twitch.tv/cyrevolt>

<https://metaspora.org/rust-wasm-electron-labortage2023.pdf>

License: CC BY 4.0 <https://creativecommons.org/licenses/by/4.0/>

