



Interactive command-line tutorials with WebAssembly

Robert Aboukhalil



sandbox.bio

sandbox.bio

https://sandbox.bio/tutorials?id=awk-intro&step=1

133% Search

Tutorials Playgrounds Log in

Filtering data

Extract columns

The first thing to know about **awk** is that it operates on **rows** and **columns**, also known as **records** and **fields**, respectively.

To extract the 3rd column of **orders.tsv**, we can use the **print** command and use **\$n** to represent the **n**th column:

```
awk '{ print $3 }' orders.tsv | head
```

Note that **print \$0** (or just **print**) refers to the entire line:

```
awk '{ print $0 }' orders.tsv | head
```

Awk also provides built-in variables: for example, **NF** (number of fields) tells you how many columns there are on the current line. In turn, **\$NF** means fetch the last column of the file:

```
guest@sandbox$ head -n 5 orders.tsv
order_id      quantity      item name      choice_description  item_price
1             1             Chips and Fresh Tomato Salsa  NULL      $2.39
1             1             Izze [Clementine]      $3.39
1             1             Nantucket Nectar [Apple] $3.39
1             1             Chips and Tomatillo-Green Chili Salsa  NULL      $2.39
guest@sandbox$ awk '{ print $3 }' orders.tsv | head
item_name
Chips
Izze
Nantucket
Chips
Chicken
Chicken
Side
Steak
Steak
guest@sandbox$ awk --version
GNU Awk 5.1.0
Copyright (C) 1989, 1991-2020 Free Software Foundation.

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program. If not, see http://www.gnu.org/licenses/.
guest@sandbox$
```

← Previous

Next →

Help 2 / 12

WebAssembly

What is WebAssembly?



```
char * hello() {  
    return "Conf42";  
}
```



```
(module  
  (table 0 anyfunc)  
  (memory $0 1)  
  (data (i32.const 16) "Conf42\00")  
  (export "memory" (memory $0))  
  (export "hello" (func $hello))  
  (func $hello (; 0 ;) (result i32)  
    (i32.const 16)  
  )  
)
```

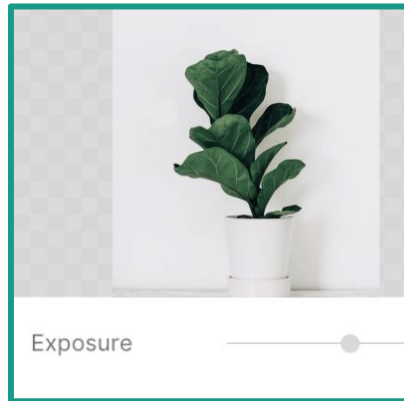
WebAssembly Text (WAT)



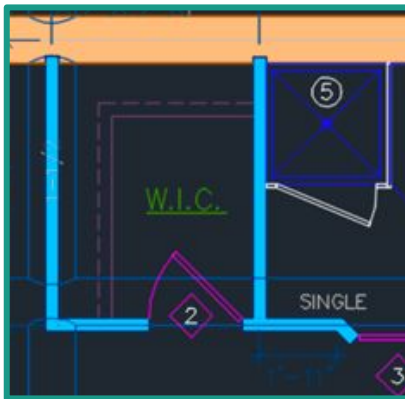
Code Reuse



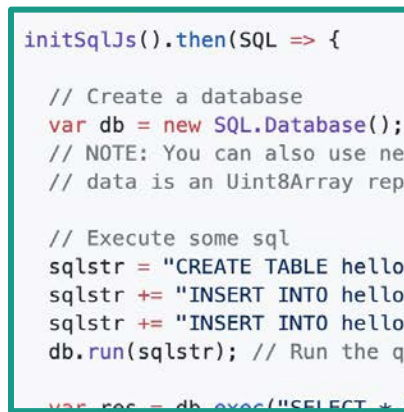
Google Earth



Figma



AutoCAD

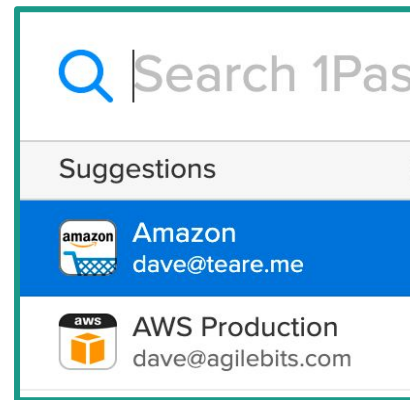


sql.js

⚡ Performance



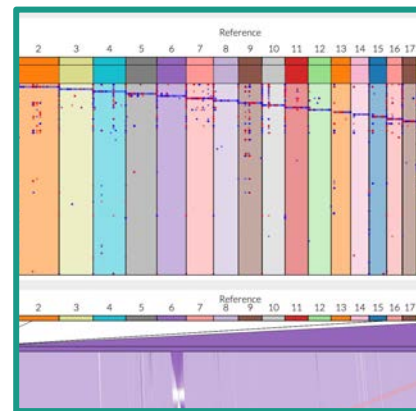
TensorFlow.js



1Password



eBay



GenomeRibbon.com



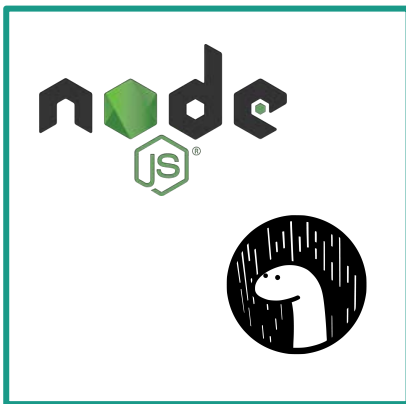
Portability



Cloudflare, Fastly



Wasmtime



Node, Deno




Wasm3

How to get started






Compile C/C++ → WebAssembly

Emscripten

 [lh3/seqtk](#) Public

Toolkit for processing sequences in FASTA/Q formats

 MIT license

 1.1k stars  291 forks

Languages

C 99.7%

Makefile 0.3%

```
$ gcc seqtk.c \  
  -o seqtk \  
  -O2 \  
  -lm \  
  -lz
```



```
$ emcc seqtk.c \  
  -o seqtk.js \  
  -O2 \  
  -lm \  
  -s USE_ZLIB=1 \  
  -s FORCE_FILESYSTEM=1
```

```
./seqtk fqchk data.fastq
```

```
Module.callMain(["fqchk", "data.fastq"])
```

Emscripten

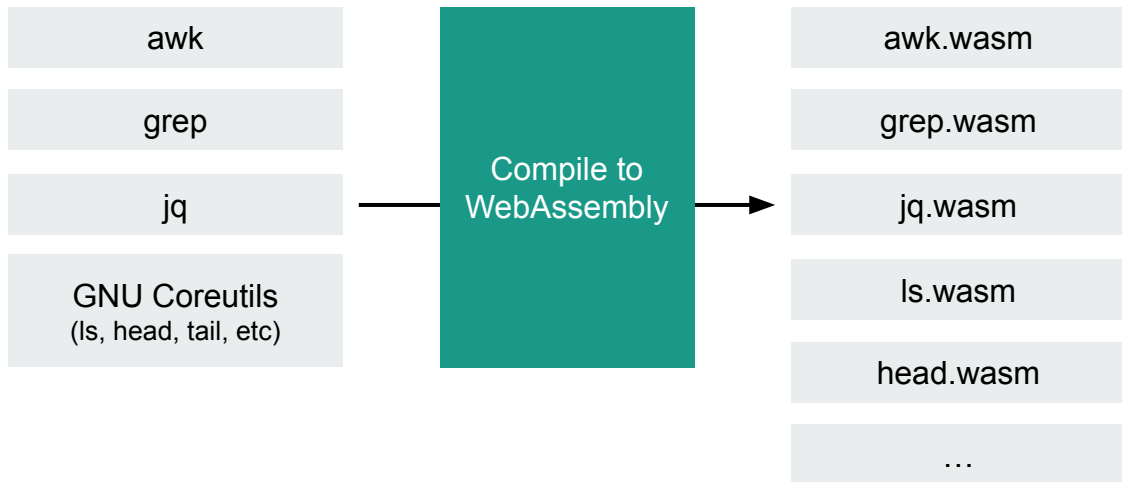
gcc	→	emcc
g++	→	em++
ar	→	emar
make	→	emmake
cmake	→	emcmake
configure	→	emconfigure

Learn more



levelupwasm.com

sandbox.bio



sandbox.bio

https://sandbox.bio/tutorials?id=awk-intro&step=1

133%

Search

Tutorials

Playgrounds

Log in

sandbox.bio

Filtering data

Extract columns

The first thing to know about **awk** is that it operates on **rows and columns**, also known as **records and fields**, respectively.

To extract the 3rd column of **orders.tsv**, we can use the **print** command and use **\$n** to represent the **n**th column:

```
awk '{ print $3 }' orders.tsv | head
```

Note that **print \$0** (or just **print**) refers to the entire line:

```
awk '{ print $0 }' orders.tsv | head
```

Awk also provides built-in variables: for example, **NF** (number of fields) tells you how many columns there are on the current line. In turn, **\$NF** means fetch the last column of the file:

← Previous

Next →

Help 2 / 12

xterm.js

```
guest@sandbox$ awk '{ print $3 }' orders.tsv | head
item_name
Chips
Izze
Nantucket
Chips
Chicken
Chicken
Side
Steak
Steak
guest@sandbox$
```

→ AST

↓

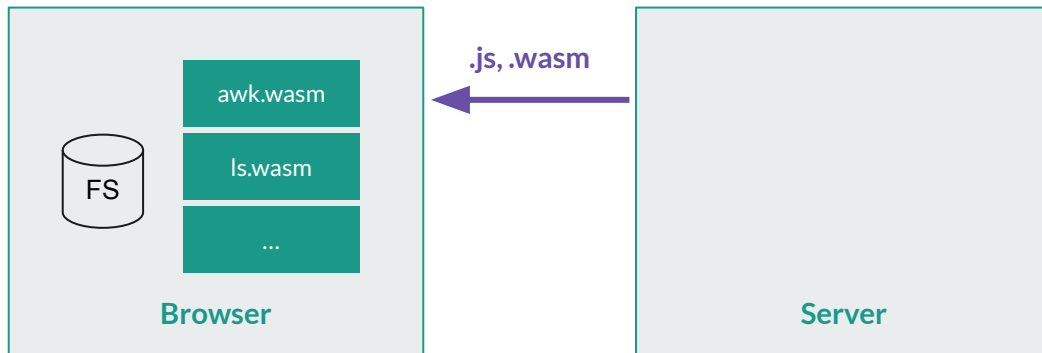
awk.wasm
head.wasm

↻

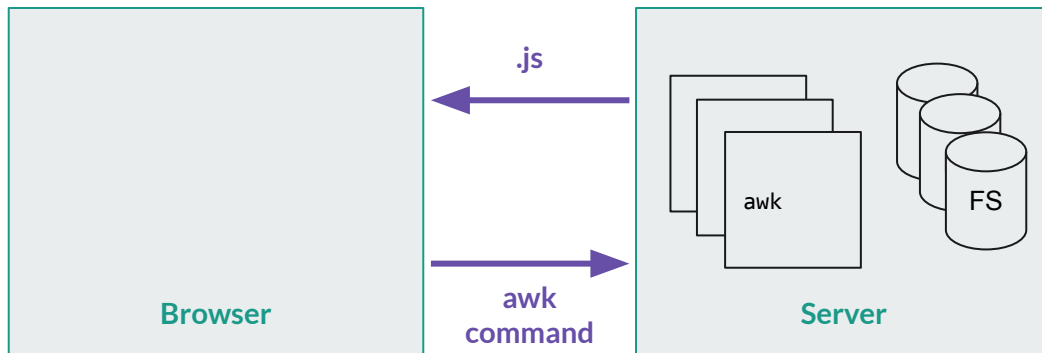
Store FS state
in IndexedDB

Why sandbox.bio uses WebAssembly

With WebAssembly



Without WebAssembly



Advantages:

- Low cost
- Highly scalable
- More secure
- More responsive
- Easier to maintain state

Disadvantages:

- Data size is limited
- Must be compilable to Wasm

When *not* to use WebAssembly

When not to use WebAssembly

Too little or too much computation (in the browser)

Frontend UI



```
fn view(&self, ctx: &Context<Self>) -> Html {
    // This gives us a component that allows us to send messages,
    let link = ctx.link(),
    html! {
        <div>
            <button onclick={link.click()} | Msg::AddOne}>{ "+1" }</button>
            <p>{ self.value }</p>
        </div>
    }
}
```

High-CPU,
high-RAM,
long running
analysis



```
1 [|||||100.0%] 5 [|||||100.0%] 13 [|||||100.0%]
2 [|||||100.0%] 6 [|||||100.0%] 14 [|||||100.0%]
3 [||||| 32.2%] 7 [|||||100.0%] 15 [|||||100.0%]
4 [||||| 32.0%] 8 [|||||100.0%] 16 [|||||100.0%]
Mem[|||||100.0%] 12 [|||||100.0%] 17 [|||||100.0%]
Swp[|||||100.0%] 18 [|||||100.0%] 19 [|||||100.0%]
GIF [|||||100.0%] 20 [|||||100.0%] 21 [|||||100.0%]
CPU: 55, 263 thr: 15 running
Average: 14.42 14.71 9.97
Time: 49 days, 04:18:55
```

Sweet Spot: audio/video processing, gaming, simulations, playgrounds, etc.

When **not** to use WebAssembly

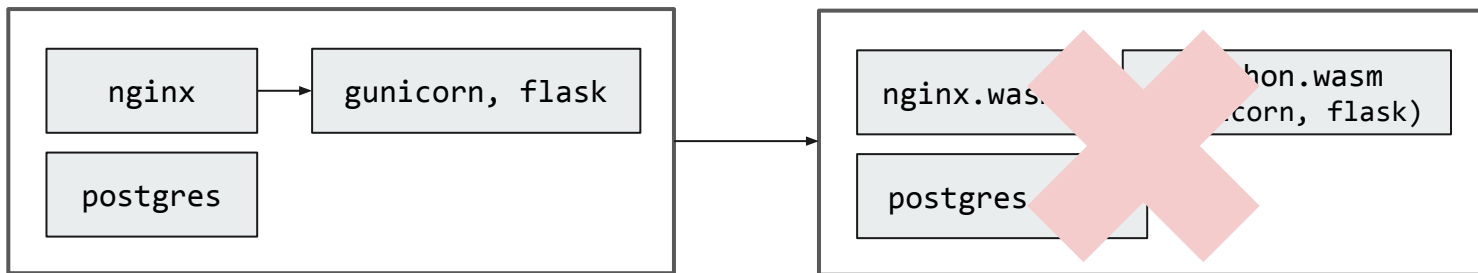
Pre-compiled tools already exist (in the browser)

Use pre-compiled tools!



When **not** to use WebAssembly

Trying to replace containers (on the server)



Simple Python web app

Sweet Spot: safely running user-provided code, edge computing

Resources

Resources

The screenshot shows the 'Data exploration with awk' tutorial page. The page has a sidebar on the left with navigation links: 'awk', 'terminal', and 'intermediate'. The main content area is titled 'Data exploration with awk by Robert Aboukhalil'. It includes a blue box with the text: 'Make sure you're comfortable with the **Terminal Basics** (e.g. `ls`, `head`, `tail`, `grep`) before going through this tutorial.' Below this, it says: 'Awk is a power tool to help you filter, extract and transform data files on the command line.' It then explains: 'It is most commonly used with tab- and comma-separated files, where the idea is to apply a certain transformation to every line (and sometimes column) of a file. As you'll see later in this tutorial, `awk` is actually a full-fledged programming language!' The tutorial continues: 'Here we'll reuse the `orders.tsv` file from the **Terminal Basics** tutorial (`head orders.tsv`), which contains take-out order data from Chipotle.' It concludes with: 'Time to wrangle some data!'. At the bottom, there are navigation buttons: 'Previous', 'Next', and 'Help 1/12'. A terminal window on the right shows the command `head orders.tsv` and its output, which is a table of order data with columns: `order_id`, `quantity`, `item_name`, and `choice_description`.

guest@andbox\$ head orders.tsv

order_id	quantity	item_name	choice_description
1	1	Chips and Fresh Tomato Salsa	NULL
1	1	Izze [Clementine]	\$3.39
1	1	Nantucket Nectar [Apple]	\$3.39
1	1	Chips and Tomatillo-Green Chili Salsa	NULL
2	2	Chicken Bowl [Tomatillo-Red Chili Salsa (Hot), (Black Beans, Rice, Cheese, Sour Cream)]	\$16.98
3	1	Chicken Bowl [Fresh Tomato Salsa (Mild), (Rice, Cheese, Sour Cream, Guacamole, Lettuce)]	\$10.98
3	1	Side of Chips	NULL
4	1	Steak Burrito [Tomatillo Red Chili Salsa, (Pajita Vegetables, Black Beans, Pinto Beans, Cheese, Sour Cream, Guacamole, Lettuce)]	\$11.75
4	1	Steak Soft Tacos [Tomatillo Green Chili Salsa, (Pinto Beans, Cheese, Sour Cream, Lettuce)]	\$9.25

Tutorials (awk, jq)

The screenshot shows the 'awk sandbox' playground page. The page has a sidebar on the left with navigation links: 'Tutorials', 'Playgrounds', and 'Log in'. The main content area is titled 'awk sandbox' and has a blue box with the text: 'Examples'. Below this, it says: 'Command Interactive'. The command area contains the following code:

```
# Skip first line (header)
NR > 1 {
  itemCount = $2
  itemName = $3

  # Track how often burritos were ordered
  # Notice that counts becomes a dictionary
  # default values of zero automatically
  if(itemName ~ /Burrito/)
    counts[itemName] += itemCount
} END {
  # Print burrito counts, split by filli
  for(item in counts)
    print(item, counts[item])
}
```

 The output area shows the following results:

```
Steak Burrito 3
Barbacoa Burrito 1
Chicken Burrito 4
Carnitas Burrito 1
```

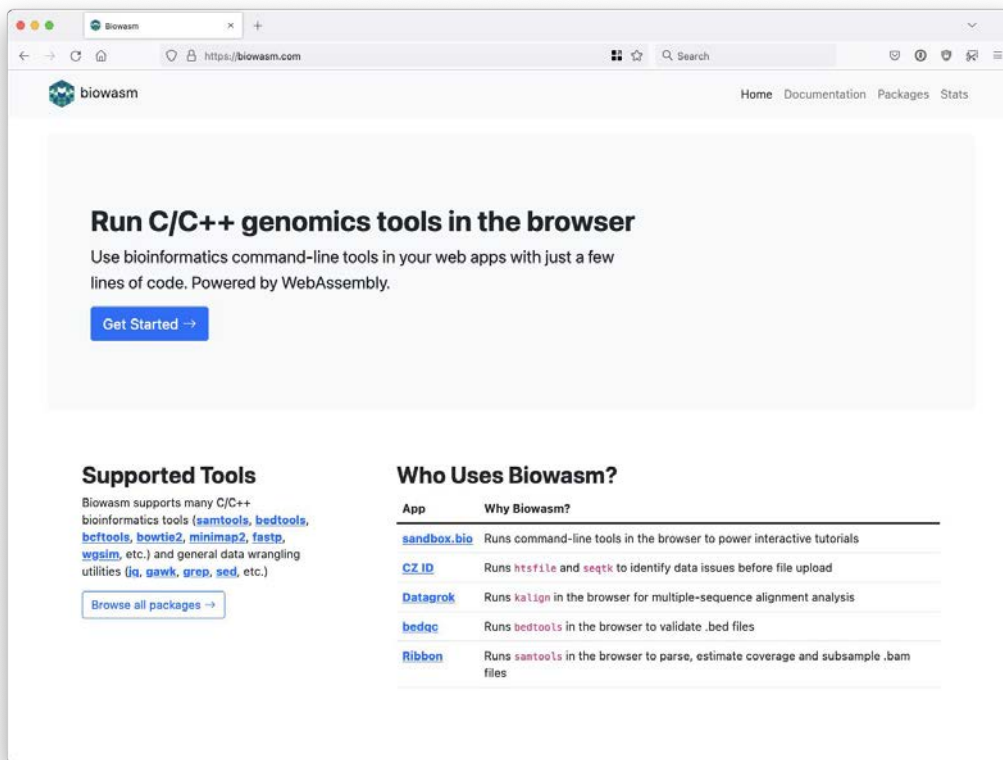
 Below the command area, there is a 'Flags' section with a dropdown menu set to 'Input Delimiter'. The flags section shows the flag `-F "\t"`. At the bottom, there is an 'Input' section with a table of order data, which is the same data as the one in the tutorial screenshot.

order_id quantity item_name choice_des

1	1	Chips and Fresh Tomato Salsa	NULL
1	1	Izze [Clementine]	\$3.39
1	1	Nantucket Nectar [Apple]	\$3.39
1	1	Chips and Tomatillo-Green Chili Salsa	
2	2	Chicken Bowl [Tomatillo-Red Chili S	
3	1	Chicken Bowl [Fresh Tomato Salsa (M	
3	1	Side of Chips	NULL
4	1	Steak Burrito [Tomatillo Red Chili S	
4	1	Steak Soft Tacos [Tomatillo Green c	
5	1	Steak Burrito [Fresh Tomato Salsa, [
5	1	Chips and Guacamole	NULL
6	1	Chicken Crispy Tacos	[Roasted Chili

Playgrounds (grep, sed, awk, jq)

Resources



Biowasm (github.com/biowasm/biowasm)
for examples of compiling complex tools to WebAssembly

Learn more @ levelupwasm.com

Level Up With WebAssembly



Robert Aboukhalil

Serverless Genomics

Using WebAssembly and Cloudflare Workers to power genomics analysis

AUGUST 28, 2019 • [4 comments](#)

**Beyond The Browser:
Getting Started With
Serverless WebAssembly**

APRIL 5, 2019 • [7 comments](#)

**How We Used WebAssembly To
Speed Up Our Web App By 20X**

**WebAssembly and SIMD: A
match made in the browser**

How SIMD makes compute-intensive tools practical on the web

**Hit the Ground Running with
WebAssembly 🚀**

**Level up command-
line playgrounds
with WebAssembly**



PerfMatters 2019



WasmSF Meetup 2019



Crossing the Wasm Chasm

levelupwasm.com/free

