
ML
CONFERENCE

# Machine Learning in the Browser


Athan Reines | @kgryte | @stdlibjs

# Overview

1. Motivation
2. JavaScript
3. Examples
4. Libraries and Frameworks
5. Web Standards
6. Tips and Tricks
7. Conclusions

# Why machine learning in the browser?

- Privacy
- Low latency
- Offline
- Power efficient
- Direct access to sensor data

# Example Datasets

- Mouse/cursor movements
- Scrolling
- Clicks
- Sensors
- Monitoring

# Use Cases

- Forecasting
- Anomalies
- Clustering

# Why JavaScript?

# Myths

- Performance
- Incompatibility

# Examples

# Anomaly Detection

# Clustering

# Regression

# Classification

# Libraries and Frameworks

# Observable

- Domain Coloring
- Routes to Chaos
- Federal Budget

# Tensorflow.js

- Predicting Strikes

# Apache Arrow

- Large Datasets

# Web Standards

# WebAssembly

A portable compilation target for the web

Allows running languages other than JavaScript within browser environments

C/C++ → IR → wasm → x86/ARM

# Two Formats

# Text Format

```
(module
  (type $0 (func (param i32 i32) (result i32)))
  (export "add" $add)
  (func $add (type $0) (param $var$0 i32) (param $var$1 i32) (result
    (i32.add
      (get_local $var$0)
      (get_local $var$1)
    )
  )
)
```

# Value Types

- i32
- f32
- i64
- f64

# Binary Encoding

```
00 61 73 6d 01 00 00 00  04 74 79 70 65 87 80 80  |.asm.....type...|
80 00 01 40 02 01 01 01  01 08 66 75 6e 63 74 69  |...@......functi|
6f 6e 82 80 80 80 00 01  00 06 6d 65 6d 6f 72 79  |on........memory|
85 80 80 80 00 80 02 80  02 01 06 65 78 70 6f 72  |...........expor|
74 86 80 80 80 00 01 00  03 61 64 64 04 63 6f 64  |t........add.cod|
65 8c 80 80 80 00 01 86  80 80 80 00 00 14 00 14  |e...............|
01 40 04 6e 61 6d 65 86  80 80 80 00 01 03 61 64  |.@.name.......ad|
64 00                                             |d.|
```

# Memory Access

- i32.load8_s (signed)
- i32.load8_u (unsigned)
- i32.load16_s
- i32.load16_u
- i32.load
- i64.load8_s
- i64.load8_u
- i64.load16_s
- i64.load16_u
- i64.load32_s
- i64.load32_u
- i64.load
- f32.load
- f64.load
- i32.store8
- i32.store16
- i32.store
- i64.store8
- i64.store16
- i64.store32
- i64.store
- f32.store
- f64.store

# Control Constructs

- nop
- block
- loop
- if
- else
- br
- br_if
- br_table
- return
- end

# Operators

- i32.add
- i32.sub
- i32.mul
- i32.div_s
- i32.div_u
- i32.rem_s
- i32.rem_u
- i32.and
- ...many more

# Advantages

- Compact
- Parsing
- Typed
- Optimization
- Deoptimization
- Lower-level
- GC
- Performance

```c
// File: dasum.c
#include <math.h>

double c_dasum( const int N, const double *X, const int stride ) {
    double sum;
    int m;
    int n;
    int i;

    sum = 0.0;
    if ( N <= 0 || stride <= 0 ) {
        return sum;
    }
    // If the stride is equal to `1`, use unrolled loops...
    if ( stride == 1 ) {
```

| Length | JavaScript | wasm | Native | Perf |
|---|---|---|---|---|
| 10 | 22,438,020 | 18,226,375 | 7,084,870 | 2.57x |
| 100 | 4,350,384 | 6,428,586 | 6,428,626 | 1.0x |
| 1,000 | 481,417 | 997,234 | 3,289,090 | 0.30x |
| 10,000 | 28,186 | 110,540 | 355,172 | 0.31x |
| 100,000 | 1,617 | 11,157 | 30,058 | 0.37x |
| 1,000,000 | 153 | 979 | 1,850 | 0.53x |

# Web Standards

- BigInt
- Value Types

# Tips and Tricks

- Avoid Built-ins
- Read the Source!
- Data Structures
  - ndarray
  - circular buffers
  - trees
- Algorithms

# Conclusions

- JavaScript
- Data Structures
- Algorithms
- Get Involved!

# Thank you!


https://github.com/stdlib-js/stdlib

https://www.patreon.com/athan

# APPENDIX

# Machine Learning

# Models

$$y = a_n x_n + a_{n-1} x_{n-1} + \cdots + a_0 x_0 + b$$

# What will be the temperature tomorrow?

$$t_{n+1} = t_n$$

$$t_{n+1} = \frac{1}{7} \sum_{i=n-6}^{n} t_i$$

$$t_{n+1} = \frac{1}{W} \sum_{i=n-W+1}^{n} t_i$$

# Training Algorithms

- **Batch**: build a model from a "batch" of data
- **Incremental**: continuously update a model as data arrives
- **Mini-batch**: hybrid of batch and incremental training.

# Real-Time Machine Learning

THE END