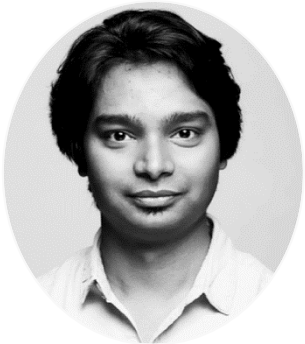# Setting up TensorFlow.js Environment

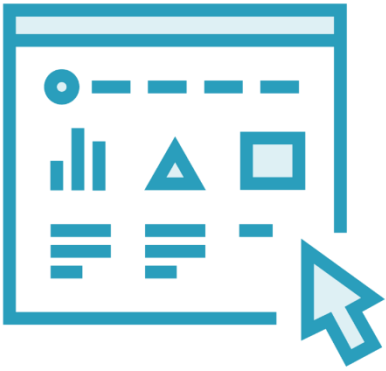**Abhishek Kumar**
DATA SCIENTIST | AUTHOR | SPEAKER

@meabhishekkumar

Supported Platforms

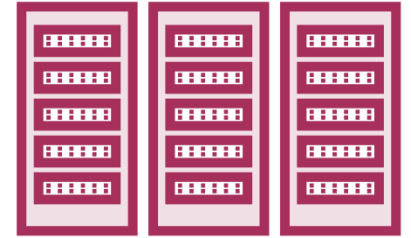**Browser**

Mobile

**Desktop and
IOT**

Server

Supported Platforms

Browser

Mobile

Desktop and
IOT

Server

Supported Platforms

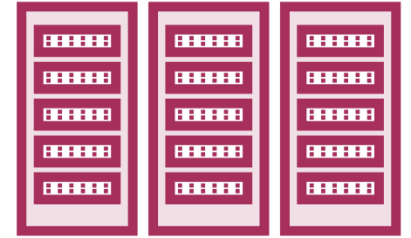**Browser**

Mobile

Desktop and IOT

Server

Overview

Browser setup

TensorFlow.js backend

Node.js setup

TensorFlow.js in Browser

```
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs"> </script>
```

Using Script Tag

Easy, but can lead to dependency management issues

Demo

Running TensorFlow.js in browser using script tag
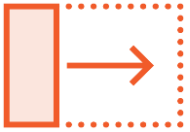
```
npm install @tensorflow/tfjs

yarn install @tensorflow/tfjs
```

Using Package Managers

Recommended approach

Why Package Managers?

Better dependency management

Dependencies mentioned in text file (e.g. package.json) that can be version controlled easily
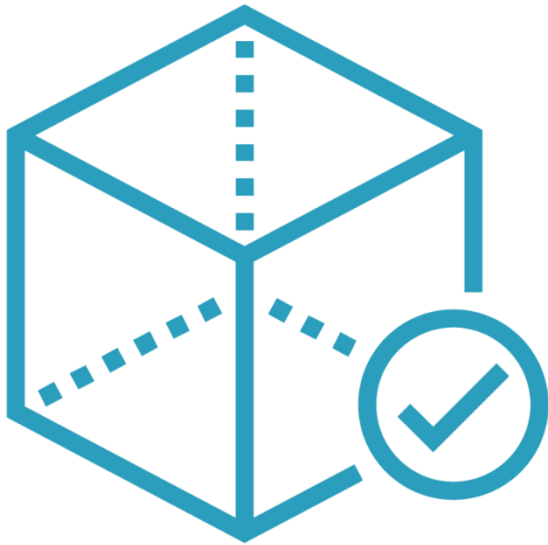
Like pip (Python), gem (Ruby)

```
npm install @tensorflow/tfjs
```

Using NPM

Common package manager for client-side and server-side dependencies (Node.js)

Web Application Bundler

Combine dependencies (JS, HTML, CSS, Others)

Benefits

- Maintain file order
- Linting
- Transpilation
- Minify

Parcel, WebPack

Demo

Running TensorFlow.js in browser using NPM and Parcel

Demo

Exploring TensorFlow.js backend

Backend

CPU

Plain vanilla JS

WebGL

GPU based

WASM

CPU acceleration
with WebAssembly

Node

TensorFlow C/C++
API

WebGPU

Near zero overhead
execution

Demo

Running TensorFlow.js in Node.js

Summary

Browser setup

- Script tag

- NPM with Parcel

Node.js setup

TensorFlow.js backend

- CPU

- WebGL

- WASM

- Node with TensorFlow C/C++ native binding

Up Next:
Understanding TensorFlow.js Core Concepts