# 1. 什么是《智能前端》？

提到《智能前端》，你能想到什么？

是根据设计稿智能生成代码的imgCook、基于GPT-3的代码生成工具Copilot、将自然语言转换为代码的NL2Code还是代码分析工具DeepCode

我觉得以上这些当然都是智能前端的缩影，那在现阶段，我们对于智能前端的认识可能还是停留在D2C或P2C阶段，从宏观的概念上讲，我认为智能前端是指从产品设计到产品发布的全链路智能化。

我的这本书，书名叫做《智能前端技术与实践》，其中，“智能”表示人工智能，具体领域是指计算机视觉，“前端”表示Web前端开发，具体领域是指网站开发、小程序开发以及扩展程序开发，我们都知道一个成熟的人工智能应用包括算法、数据以及工程（即大前端）三个方面，因此，在人工智能时代，前端智能化是Web前端领域的重要发展趋势之一。

从技术栈的角度来说，本书包含计算机视觉和Web前端开发两大领域，使用到的工具是TensorFlow.js，本书的所有案例都被部署在我的个人自建平台上，包括个人网站、微信小程序以及Google扩展程序；

本书以前端智能化为大背景，核心技术栈包括Web前端开发和深度学习两大领域，使用到的工具是TensorFlow.js，从宏观上看，全书可以分为3部分，共计12章、13个案例，第一部分（第1～5章）涵盖本书案例所依赖的开发环境、前端开发基础知识、TensorFlow 2基础知识、深度学习基础知识、TensorFlow.js框架以及常见的轻量级神经网络算法，包括MobileNet V1/V2/V3、ShuffleNet V1/V2、SqueezeNet和Xception；第二部分（第6～11章）在第一部分的基础上完成了恶意文本检测、线性回归、逻辑回归、XOR问题、人体姿态检测、目标检测、OCR识别等13个案例；第三部分（第12章）涵盖Google官方开发的一些有趣的人工智能应用，如基于RNN和LSTM的股价预测、Google拉面馆、猜画小歌、3D人脸涂鸦、Invisibility Cloak以及Teachable Machine等，本书的所有案例都被部署在我的个人自建平台上，包括个人网站、微信小程序以及Google扩展程序，大家可以在我的平台上便捷的体验到人工智能应用的魅力～

我希望本书能够帮助相关从业者或任何对深度学习和前端开发感兴趣的朋友快速入门，进而促进复杂人工智能在移动端设备或嵌入式设备上的应用。

# 2. 为什么是TensorFlow.js？

ONNX：Open Neural Network Exchange，是一种针对机器学习所设计的开放式的文件格式，用于存储训练好的模型，是一个深度学习模型标准，该标准可以使深度学习模型在不同的深度学习框架之间平滑的进行迁移，那目前支持ONNX的深度学习框架包括Caffe2、亚马逊MXNet、英伟达 TensorRT、微软CNTK、Paddle、PyTorch、TensorFlow

在浏览器端运行深度学习模型的几种方法？

1. deeplearn.js ➡ TensorFlow.js Core
2. paddlejs ➡ paddlejs-converter
3. onnx.js ➡ 微软的开源项目onnx.js，用来将torch/TensorFlow模型转换为onnx格式，然后在浏览器端通过onnx.js执行。生态混乱，在处理模型时总是出现各种各样的问题，其次很长时间没有更新了（最近一次更新是readme.md文档，4个月前，核心文档都是至少12个月之前）
4. pytorchjs/torchjs ➡️官方不支持，GitHub热度低

复杂方法：

1. pytorch ➡ onnx ➡ TensorFlow ➡ TensorFlow.js
2. Caffe/TensorFlow/ONNX/PyTorch ➡ Paddle ➡ paddlejs-converter

华为 ➡ MindSpore

旷视科技 ➡ MegEngine

2018年3月，TensorFlow宣布重大更新：增加支持JavaScript，并推出开源库TensorFlow.js，用户可以完全在浏览器定义、训练和运行机器学习模型。

在Web端通过JavaScript能够使用GPU进行计算的方式主要是利用WebGL或WebGPU的计算特性，利用WebGL调用GPU进行并行化计算的特性充分发挥GPU的并行计算能力，同时结合WebWorker多线程技术，提高大数据量结果的输出速度。

TensorFlow.js支持多种backend用来实现张量存储和数学运算，它会根据当前环境帮助我们自动选择最佳的backend。

WebGL：当前适用于浏览器功能最强大的backend，其速度比普通的CPU快100倍，这是因为它会将张量存储为WebGL纹理，所有的数学运算都将在WebGL Shader中实现。

内存管理：如果使用WebGL作为backend，就需要进行显示的内存管理，浏览器是不会自动回收WebGL Texture的垃圾，因此，在我的代码中大家会看到tf.tidy或tf.dispose等一些方法用来清理执行函数时可能产生的一些局部变量。

精度：机器学习模型中使用到的数字精度为32位，而在移动端设备上，WebGL可能仅支持16位，因此在使用之前，我们需要检查当前设备的精度，从而避免当为移动端设备移植模型时会产生的精度问题。

Node.js：使用TensorFlow C API进行加速，这种情况下，可以使用计算机的硬件进行加速，比如cuda

WASM：一种基于Web的二进制格式，被Chrome、Safari、Firefox、Edge等主流浏览器支持，可以实现CPU加速，替代普通的CPU和WebGL，速度快于JavaScript，这是因为浏览器加载、解析、执行wasm二进制文件的速度要比JS软件包的速度快；WebGL对于大多数模型而言速度均快于WASM（posenet 4.5MB、bodypix4.6MB、MobileNet V2 13MB），但针对小模型（FaceMesh 2.8MB、BlazeFace 0.4MB），其性能会优于WebGL，这是因为WebGL Shader存在一定的开销成本。

CPU：性能最低且最简单的backend，所有运算均在JS中完成，阻塞页面线程。

# 如何高效阅读本书？

如果你是Web前端开发工程师、前端AI工程师、机器学习算法工程师或计算机相关专业以及任何对前端开发和深度学习感兴趣的人，那么我非常推荐你阅读本书，如果你是Web前端和深度学习两大领域的小白，那你也完全可以阅读本书，这是因为在本书的基础知识篇中，我对书中案例的开发环境、相关基础知识做了非常详细的介绍，同时为了帮助大家更好的理解书中的内容，我将所有案例部署在了我的个人自建平台上，并将书中的所有代码进行了开源，希望这本书能够帮助到所有对智能前端开发感兴趣的朋友。

我整理了本书的核心技术栈和思维导图帮助大家更快的理清本书的知识脉络

# 沉浸式案例体验

个人网站：预训练模型加载、你画我猜（MNIST）

微信小程序：人体姿态检测、目标检测、语音识别

扩展程序：hahaOCR

Face Touch Monitor：https://facetouchmonitor.com/

Teachable Machine：https://teachablemachine.withgoogle.com/

TensorFlow.js Gallery