Sensors Converge

Grow Sensor Adoption with Standard JavaScript Drivers

June 20–22, 2023 | Santa Clara, CA

#SensorsConverge

Who am I?

moddable

Andy Carle, PhD

Co-Founder, Moddable Tech, Inc.
Member, Ecma International Technical Committee 53
Contributor to the ECMA-419 ECMAScript embedded systems API specification

Sensors Converge

Why Do Sensor Drivers Matter?

The availability of high-quality, easy-to-use, full-featured drivers is a crucial consideration when selecting which sensor to use in an IoT product.

Basic sample code showing how to use a component in a generic context is not enough. Fully-realized, production-ready drivers that are portable across silicon solutions are a massive competitive advantage for securing design wins in IoT product development.

Sensor Manufacturers

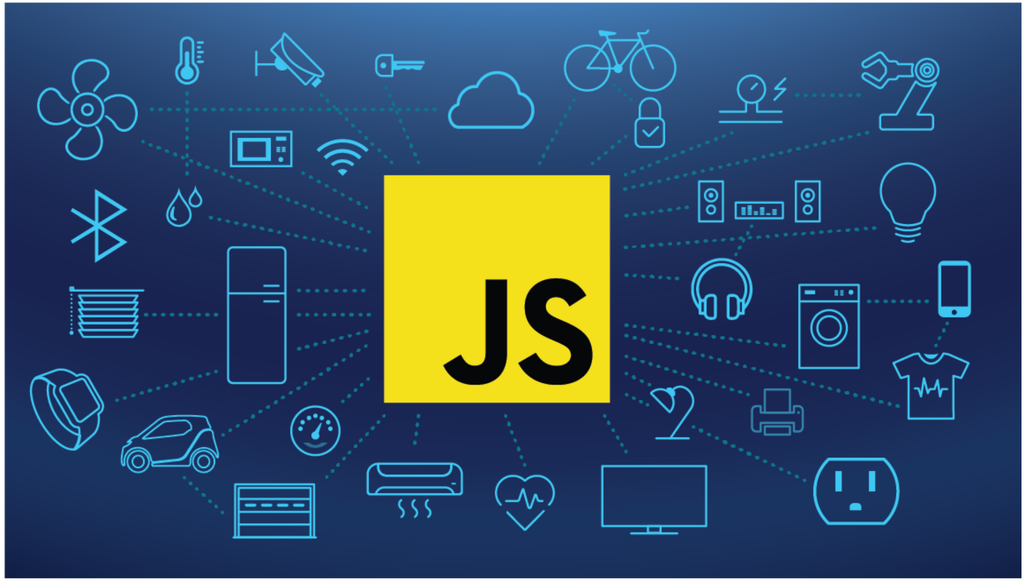
Must provide these drivers and make them easily discovered by designers who are selecting component to secure design wins.

IoT Product Developers

Seek these drivers while selecting components and building initial prototypes.

Why JavaScript for Embedded Systems?

- **JavaScript is rapidly becoming the *lingua franca* of product development across all platforms**
- **JavaScript-based products are:**
  - simple to write and maintain
  - easily ported across silicon solutions
  - inherently reliable & secure
  - built on open standards
  - a natural fit for products with cloud connectivity and/or displays
- **The JavaScript ecosystem accommodates developers with many different needs and skills:**
  - TypeScript
  - Node-RED / Node-RED MCU Edition
- **The Moddable SDK is an open source platform for IoT product development with hundreds of useful examples and modules for building product firmware**



ECMA-419: ECMAScript embedded systems API specification

A JavaScript standard for resource-constrained microcontrollers and other embedded systems.

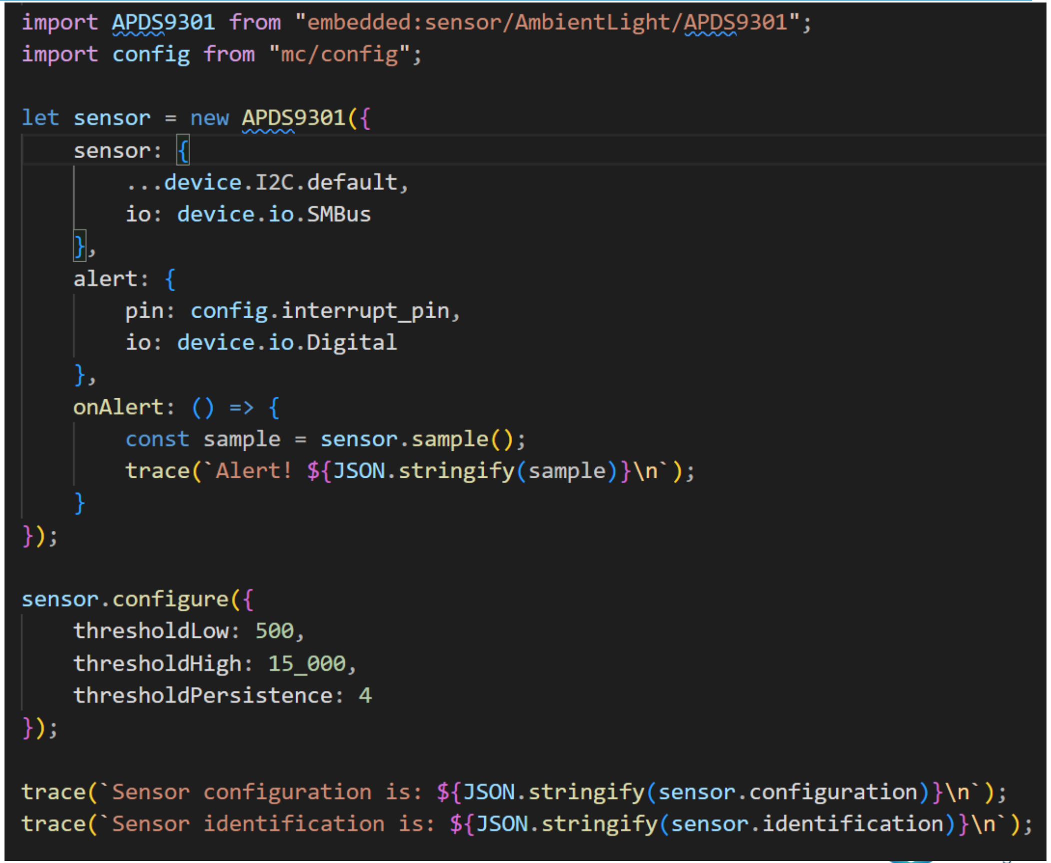
Includes standard APIs for:

- Interacting with MCU I/O
- Configuring and sampling generic sensors
- Working with dozens of specific sensor types
- Controlling peripherals & displays
- Networking connections & protocols

https://419.ecma-international.org/

Sensor Class Pattern

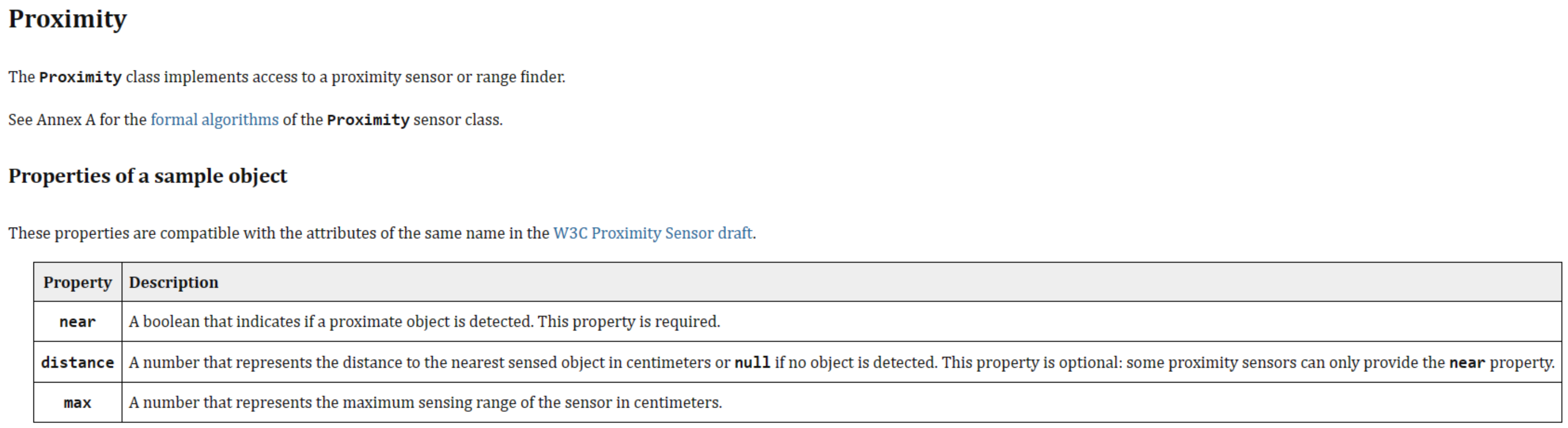
- **A JavaScript API with 3 fundamental methods, an error-handling callback, and supporting details**
- **Methods:**
  - constructor
  - configure
  - sample
- **Callback:**
  - onError
- **The entire Sensor Class Pattern is only about a page long**
- **Compatible with the W3C Generic Sensors API for the web**



```javascript
import APDS9301 from "embedded:sensor/AmbientLight/APDS9301";
import config from "mc/config";

let sensor = new APDS9301({
    sensor: {
        ...device.I2C.default,
        io: device.io.SMBus
    },
    alert: {
        pin: config.interrupt_pin,
        io: device.io.Digital
    },
    onAlert: () => {
        const sample = sensor.sample();
        trace(`Alert! ${JSON.stringify(sample)}\n`);
    }
});

sensor.configure({
    thresholdLow: 500,
    thresholdHigh: 15_000,
    thresholdPersistence: 4
});

trace(`Sensor configuration is: ${JSON.stringify(sensor.configuration)}\n`);
trace(`Sensor identification is: ${JSON.stringify(sensor.identification)}\n`);
```

Sensor Classes

- **Define common sensor capabilities for specific sensor categories**
- **Sensor capabilities not supported by the Sensor Class may be added in well-defined ways**



Proximity

The **`Proximity`** class implements access to a proximity sensor or range finder.

See Annex A for the formal algorithms of the **`Proximity`** sensor class.

Properties of a sample object

These properties are compatible with the attributes of the same name in the W3C Proximity Sensor draft.

Property	Description
`near`	A boolean that indicates if a proximate object is detected. This property is required.
`distance`	A number that represents the distance to the nearest sensed object in centimeters or **`null`** if no object is detected. This property is optional: some proximity sensors can only provide the **`near`** property.
`max`	A number that represents the maximum sensing range of the sensor in centimeters.

Why Build & Use Conformant Sensor Drivers?

Sensor drivers that implement an ECMA-419 Sensor Class can be efficiently integrated into product prototypes and easily interchanged, while still exposing product-specific features.

Sensor Manufacturers

Create a plug-and-play sensor driver to open the door to design wins at the prototyping stage. Exceptional hardware features can be easily used by developers and can help prototyping wins carry through to production.

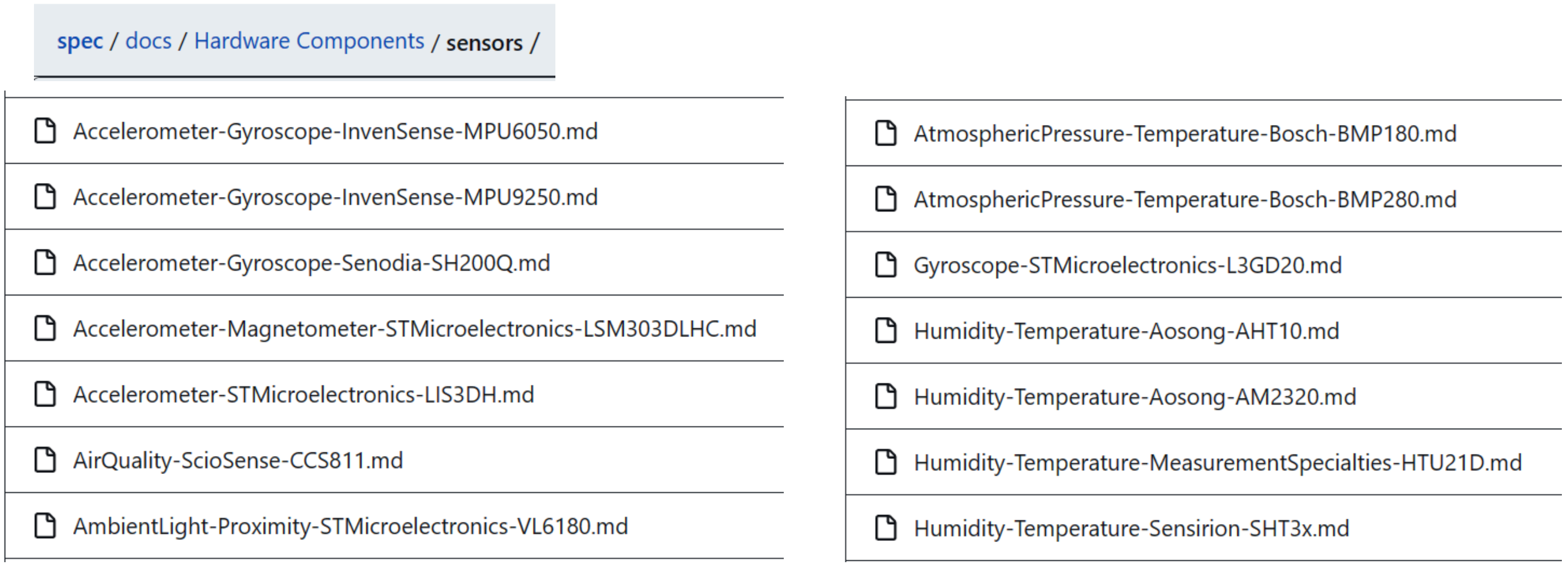
IoT Product Developers

Can rapidly prototype with many different, interchangeable components to experiment with each part's strengths and weaknesses, including accuracy, reliability, and special features.

Can more easily change components later to adjust to supply constraints.

Class Specifications for Embedded Hardware Components

A publicly-available, royalty-free repository of ECMAScript class definitions for specific embedded hardware components.

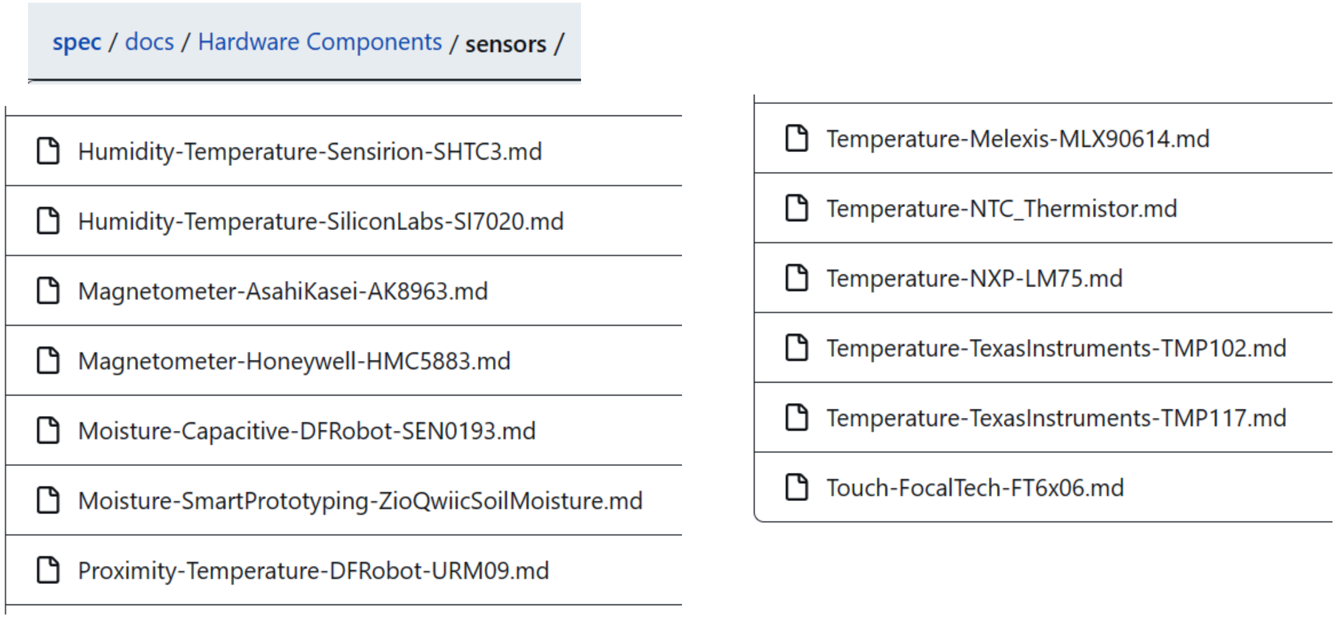


spec / docs / Hardware Components / sensors /

- Accelerometer-Gyroscope-InvenSense-MPU6050.md
- Accelerometer-Gyroscope-InvenSense-MPU9250.md
- Accelerometer-Gyroscope-Senodia-SH200Q.md
- Accelerometer-Magnetometer-STMicroelectronics-LSM303DLHC.md
- Accelerometer-STMicroelectronics-LIS3DH.md
- AirQuality-ScioSense-CCS811.md
- AmbientLight-Proximity-STMicroelectronics-VL6180.md
- AtmosphericPressure-Temperature-Bosch-BMP180.md
- AtmosphericPressure-Temperature-Bosch-BMP280.md
- Gyroscope-STMicroelectronics-L3GD20.md
- Humidity-Temperature-Aosong-AHT10.md
- Humidity-Temperature-Aosong-AM2320.md
- Humidity-Temperature-MeasurementSpecialties-HTU21D.md
- Humidity-Temperature-Sensirion-SHT3x.md

Class Specifications for Embedded Hardware Components

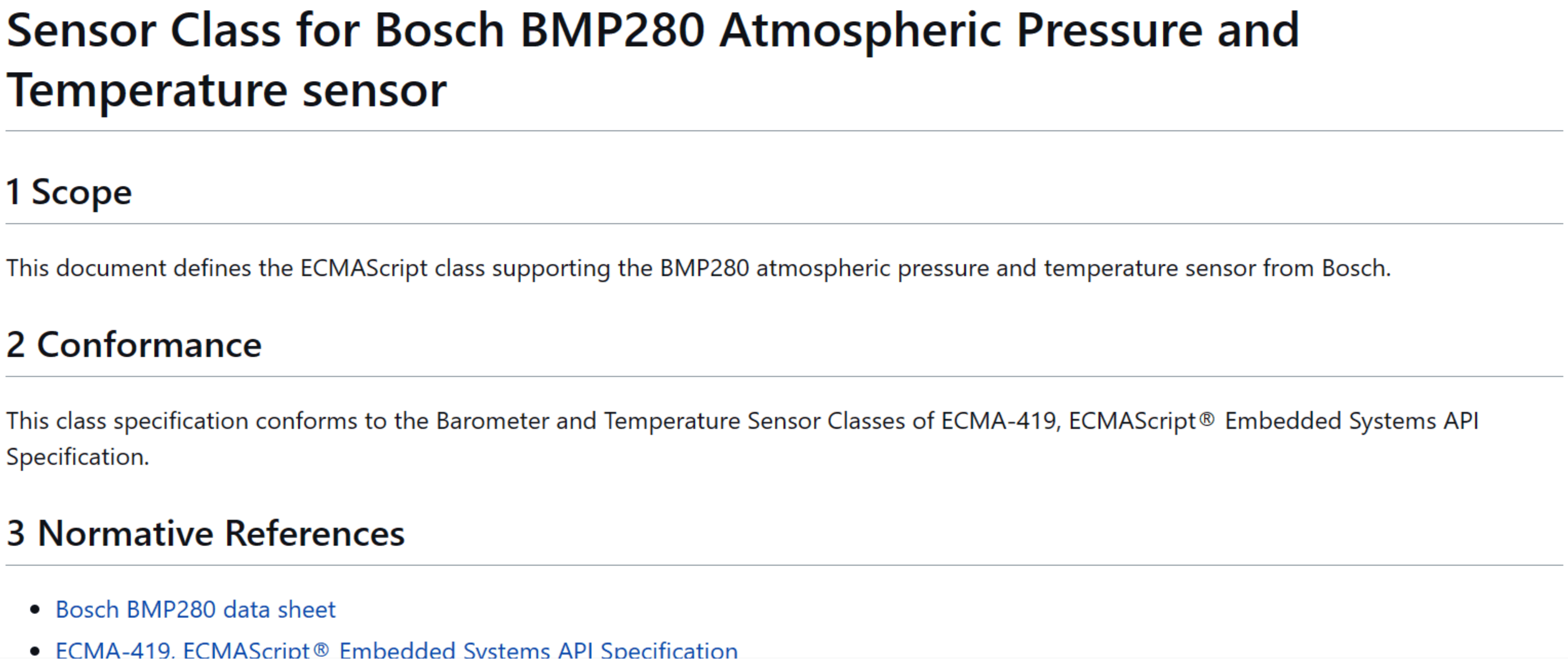
A publicly-available, royalty-free repository of ECMAScript class definitions for specific embedded hardware components.



spec / docs / Hardware Components / sensors /

- Humidity-Temperature-Sensirion-SHTC3.md
- Humidity-Temperature-SiliconLabs-SI7020.md
- Magnetometer-AsahiKasei-AK8963.md
- Magnetometer-Honeywell-HMC5883.md
- Moisture-Capacitive-DFRobot-SEN0193.md
- Moisture-SmartPrototyping-ZioQwiicSoilMoisture.md
- Proximity-Temperature-DFRobot-URM09.md
- Temperature-Melexis-MLX90614.md
- Temperature-NTC_Thermistor.md
- Temperature-NXP-LM75.md
- Temperature-TexasInstruments-TMP102.md
- Temperature-TexasInstruments-TMP117.md
- Touch-FocalTech-FT6x06.md

Class Specifications for Embedded Hardware Components



Sensor Class for Bosch BMP280 Atmospheric Pressure and Temperature sensor

1 Scope

This document defines the ECMAScript class supporting the BMP280 atmospheric pressure and temperature sensor from Bosch.

2 Conformance

This class specification conforms to the Barometer and Temperature Sensor Classes of ECMA-419, ECMAScript® Embedded Systems API Specification.

3 Normative References

- Bosch BMP280 data sheet
- ECMA-419, ECMAScript® Embedded Systems API Specification

Class Specifications for Embedded Hardware Components

A publicly-available, royalty-free repository of ECMAScript class definitions for specific embedded hardware components.

Sensor Manufacturers

Contribute sensor driver documentation to the repository to make sensor driver modules easily discovered and aid in the integration of sensor drivers into product prototypes.

IoT Product Developers

Browse the repository to find sensor drivers that fit product design needs and conform to the relevant ECMA-419 Sensor Class.

https://github.com/EcmaTC53/spec/tree/master/docs/Hardware%20Components

Get Involved: Sensor Manufacturers

1. **Design a sensor class API for your sensor that conforms with application ECMA-419 classes**
   - Browsing the Hardware Components repository to see examples will ease this process
2. Develop a JavaScript sensor driver module for your component
   - Start from existing sensor driver implementations in the Moddable SDK to inherit reliable development patterns
3. **Agree to the Ecma International Policy on Submission, Inclusion and Licensing of Software**
   - https://www.ecma-international.org/wp-content/uploads/Ecma_Policy_on_Submission_Inclusion_and_Licensing_of_Software.pdf
4. **Open a Pull Request on the EcmaTC53/Spec repository to publish your document and link to your implementation**
   - https://github.com/EcmaTC53/spec/pulls

Bonus: Join Ecma International & TC53 to help shape the future of sensors for Embedded JavaScript

Get Involved: Product Developers

Conventional Developers

1. Download a JavaScript runtime for embedded, such as the Moddable SDK

Low-Code Developers

1. Download Node-RED MCU Edition

2. Experiment with basic sensor-based examples on the microcontroller of your choice
3. Browse the Class Specifications for Embedded Hardware Components to find a sensor that matches your product needs
4. Order hardware samples and experiment with the open source sensor driver using the real component
5. Freely change components using the same application code by simply swapping sensor drivers

Case Study: Spraying Systems Sustainability Monitor

- **Interchangeable ECMA-419-conformant sensors**
- **Beautiful touchscreen interface**
- **Cloud connectivity**

Sensors Converge

Conclusion

- Providing real, portable, production-ready drivers for sensors is a massive competitive advantage for securing IoT design wins at the prototyping stage
- Working in the JavaScript ecosystem provides access to a large developer pool and diverse tools:
  - Moddable SDK: https://www.moddable.com/
  - Node-RED MCU Edition: https://github.com/phoddie/node-red-mcu
  - TypeScript for enterprise development: https://www.typescriptlang.org/
- The ECMA-419 Specification provides a target for driver development that allows a component to be easily “dropped in” to a product prototype while still exposing unique features of the component
  - https://419.ecma-international.org/
- The Class Specifications for Embedded Hardware Components repository is a marketplace for components and drivers that can be used for new IoT product designs
  - https://github.com/EcmaTC53/spec/tree/master/docs/Hardware%20Components

Come chat with me in Booth 802 to learn more!