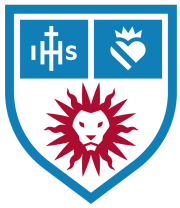


Loyola Marymount University

Software Design Specification

Open Source React Libraries for Mobile Browser Sensor Development

Revision 2.0

Table of Contents

Introduction **1**
- Purpose 1
- System Overview 1
- Definitions and Acronyms 1

Design Considerations **1**
- Assumptions 1
- Constraints 1
- System Environment 2
- Design Methodology 2
- Risks and Volatile Areas 2

Architecture **3**
- Overview 3
- Use of React Hooks 3

Software Design **5**
- Gyroscope 5
- Absolute Orientation 6
- Accelerometer 7
- Ambient Light Sensor 8
- Gravity 9
- Linear Acceleration 10
- Magnetometer 11

Appendix A: Project Timeline **12**

Revision History

Version	Name	Reason For Changes	Date
1.0	*Masao Kitamura*	*Initial Revision*	*2/21/2022*
2.0	*Masao Kitamura*	*Final Revision*	*4/29/2022*

1. Introduction

1.1 Purpose

This design will detail the implementation of the requirements as defined in the Open Source React Libraries for Mobile Browser Sensor Development capstone project.

1.2 System Overview

This project extends the functionality of the Open Source React Libraries for Mobile Browser Sensor Development capstone project. Additional libraries will be published to the Node Package Manager (online) which is the central location for all public Node and React modules.

The main demo which demonstrates the functionality of all newly-created React hook sensors can be access here:

https://github.com/masaok/react-mobile-sensors-demo

Each React hook sensor has been published to the Node Package

- Absolute Orientation Sensor
  - https://www.npmjs.com/package/react-hook-absolute-orientation
- Accelerometer
  - https://www.npmjs.com/package/react-hook-accelerometer
- Ambient Light Sensor
  - https://www.npmjs.com/package/react-hook-ambient-light
- Gravity Sensor
  - https://www.npmjs.com/package/react-hook-gravity
- Linear Acceleration
  - https://www.npmjs.com/package/react-hook-linear-acceleration
- Magnetometer
  - https://www.npmjs.com/package/react-hook-magnetometer
- Gyroscope
  - https://www.npmjs.com/package/react-hook-gyroscope
- Geolocation
  - https://www.npmjs.com/package/react-hook-geolocation

1.3 Definitions and Acronyms

React.js – Javascript framework for quickly building web frontends
Node.js - a Javascript runtime built on Chrome's V8 Javascript engine
Chrome - a cross-platform web browser created by Google
Frontend - The user interface for a web software application
Javascript – A programming language used in both frontend and backend development.

Backend - Any part of an application that does not involve the frontend, typically, the API and data access layers
API - Application Programming Interface, a set of definitions and protocols for building application software
React Hook - a React feature which allows a developer to use state and other React features without creating a class

2. Design Considerations

All design considerations were handled in React Libraries Phase 1.

2.1 Assumptions

The Mozilla API is up-to-date with the latest documentation and compatibility requirements for the Javascript sensors for mobile browsers.

2.2 Constraints

This project is constrained by the compatibility of sensors for each mobile manufacturer and browser.

2.3 System Environment

The React libraries will be stored on the NPM servers remotely.

A React developer will need a workstation with Node.js installed. After creating a working React hook with each mobile browser sensor, the hook repository will need to be published to NPM.

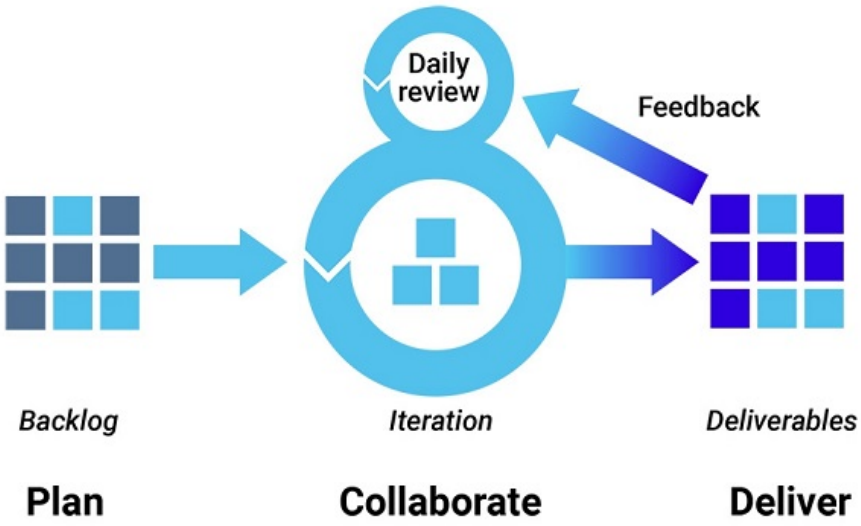
2.4 Design Methodology

The agile development methodology was used in the development of the React libraries.

The agile development methodology includes planning, collaboration, and delivery.

Planning consists of a backlog of tasks which are then iterated upon in the collaboration phase.

After the iteration phase, the deliverables are then evaluated and feedback is given in a daily review. Development in the iteration phase is then resumed again completing the loop back into deliverables. This loop continues as long as is needed to finish the project.



Source: https://www.synopsys.com/blogs/software-security/top-4-software-development-methodologies/

2.5 Risks and Volatile Areas

Several risk factors have been identified in the development of this project.

First, there is a strict deadline to complete this project by the end of April 2022 due to the academic calendar. Although ongoing development can continue, the basic libraries must work by the end of the semester. This risk will be mitigated through setting realistic deadlines and prevention of scope creep.

Second, since I will be the sole developer on the project, I will need to balance other commitments including work, family, diet, and getting enough rest. Since these other commitments can be unpredictable, it is an additional risk to the timely completion of the project.

Third, although unlikely, the APIs to the browser-based motion sensors may change over time. If this happens, the React libraries will need to be adjusted to accommodate any changes to the sensor APIs.

3. Architecture

The architecture provides the top level design view of a system and provides a basis for more detailed design work

Provide or reference a detailed description and diagrams of the architecture..

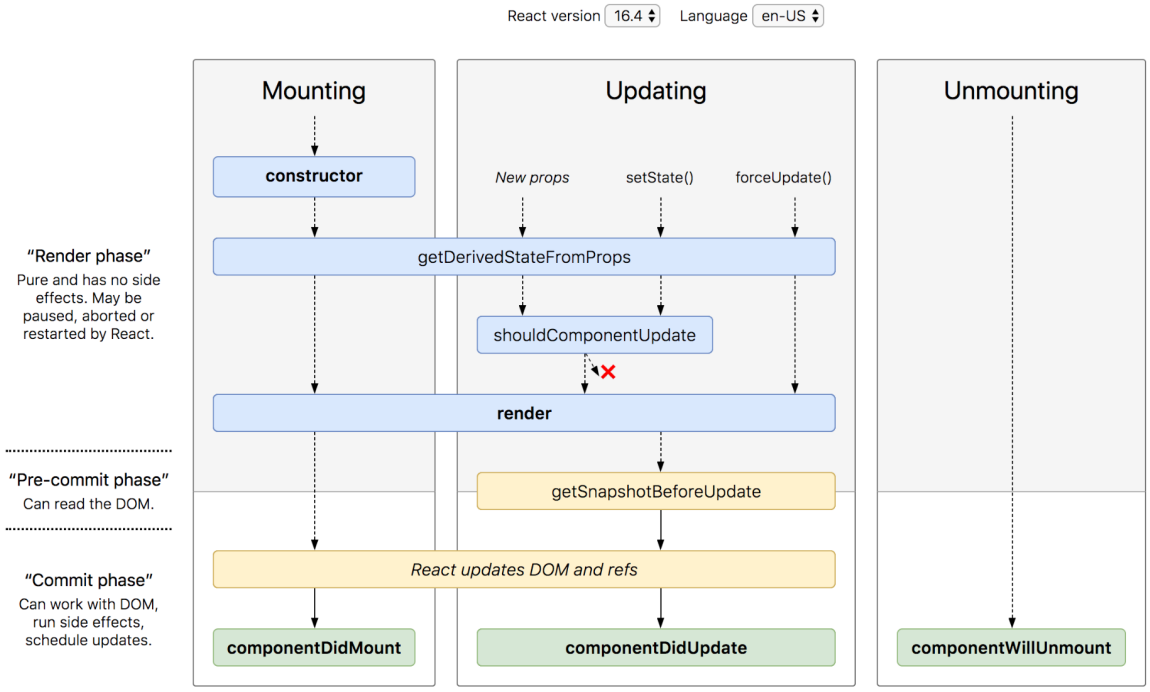
3.1 Overview

This section provides a high level overview of the structural and functional decomposition of the system.

3.2 Use of React Hooks

React Hooks is the modern method of managing state without needing to create classes in a React project.

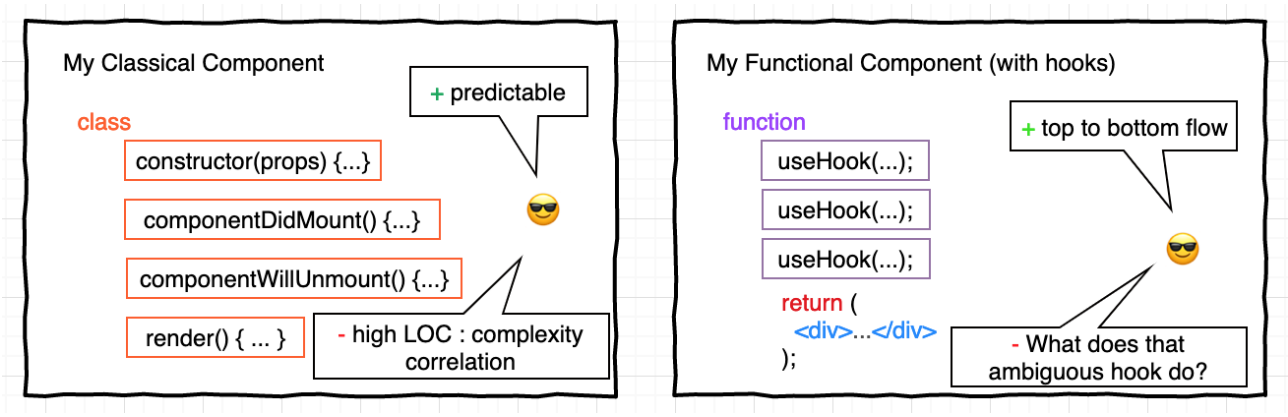
The older way of creating React classes is shown below:



Source: https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

However, given the introduction of functional components and React Hooks in February of 2019, this has

The two styles of components are shown in the diagram below:



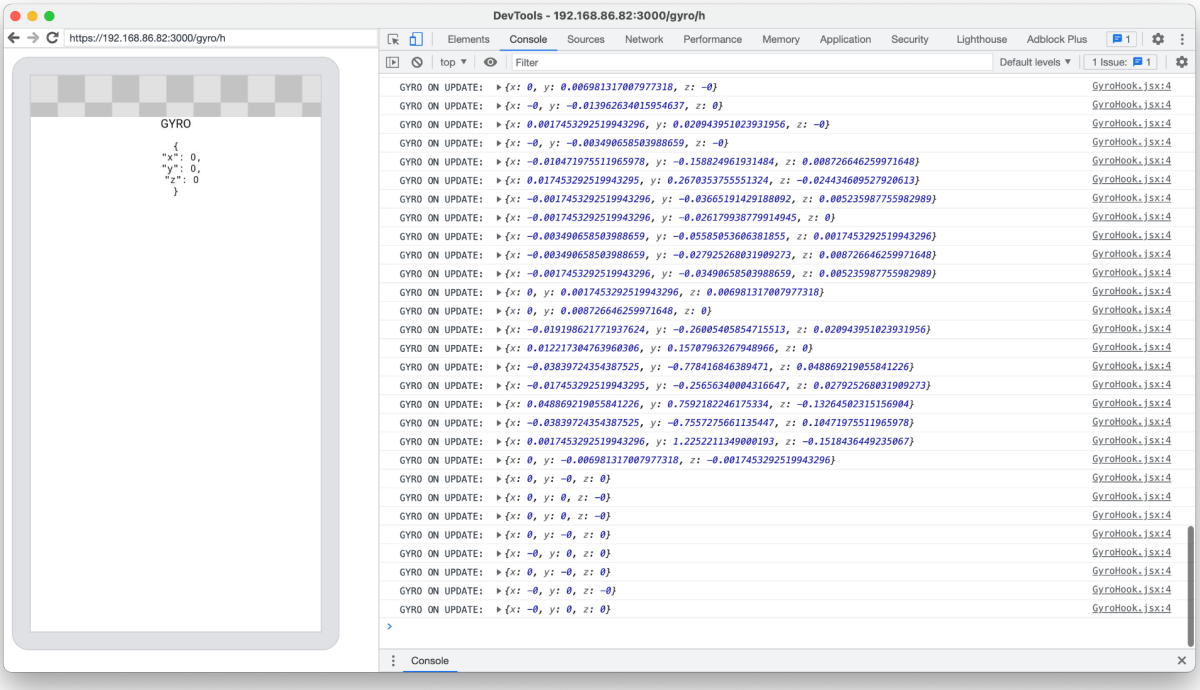
Source: https://medium.com/@bryanmanuele/react-hooks-best-practices-a-shift-in-mindset-8fd0e58e4b0b

On the left, the older style of React components is shown with the most common functions included in most classes.

On the right, the newer style of functional components with hooks is displayed in abbreviated fashion. The goal of this project will be to create React Hooks that give access to the motion sensor APIs in mobile browsers.

4. Software Design

4.1 Gyroscope



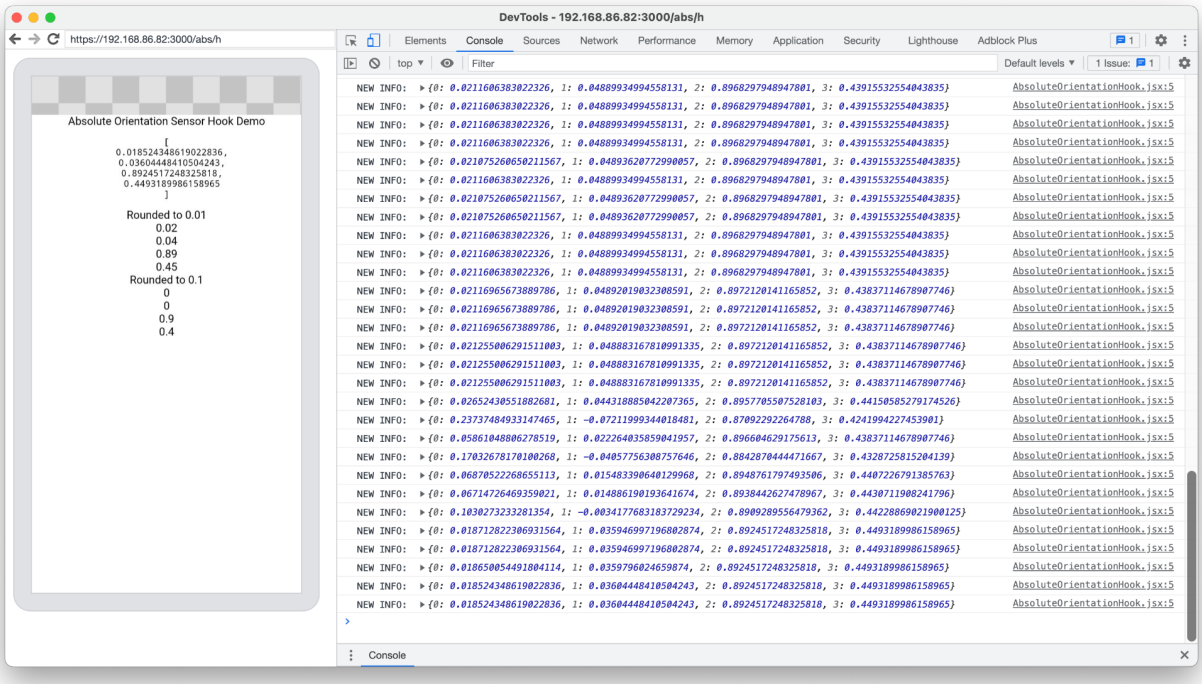
The Gyroscope sensor provides the angular velocity of the mobile device along all three axes.

Internally to the React hook, the code will save the state of sensor readings in a variable.

Each time the Gyroscope sensor updates on a given frequency, the hook will update its state and give the latest readings back to the caller.

Upon error, the hook will set all variables to null and log the error to the console.

4.2 Absolute Orientation



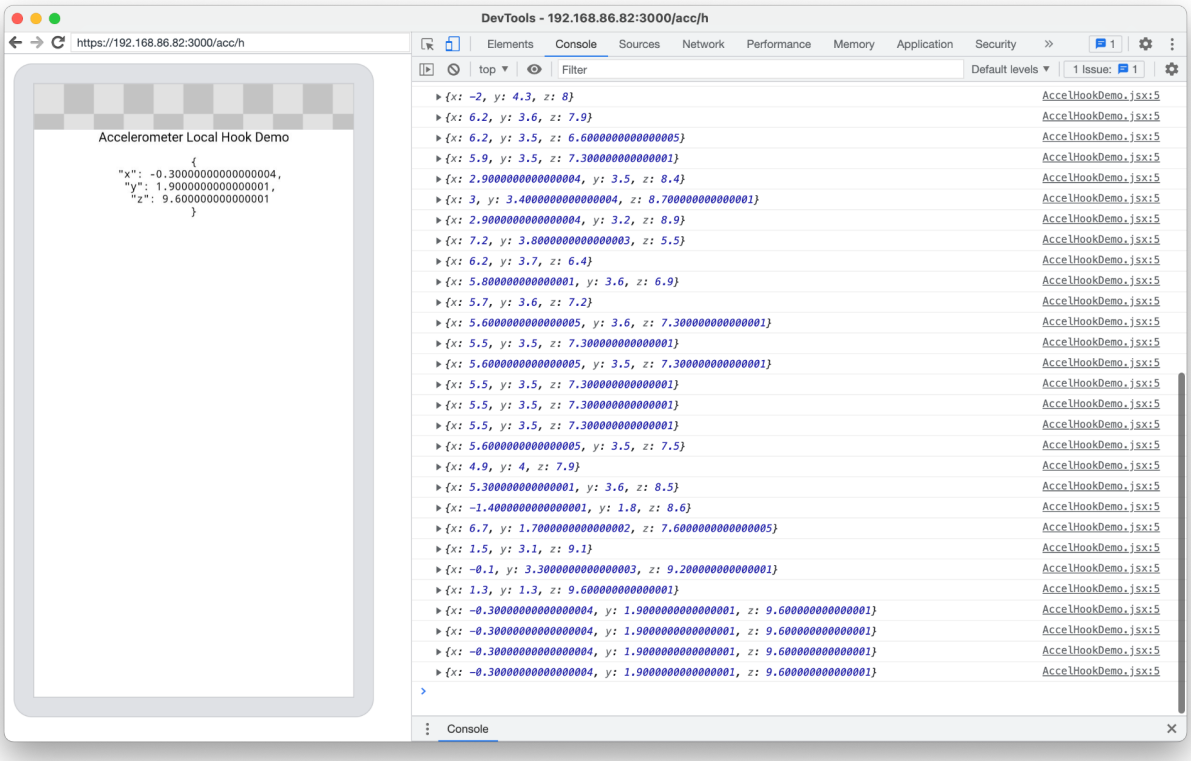
The Absolute Orientation sensor provides a device's physical orientation in relation to a 3D Cartesian coordinate system. A Cartesian coordinate system is defined as three perpendicular lines that intersect at a point of origin.

The React hook will store the sensor reading in a state variable which represents a quaternion. A quaternion is a quotient of two directed lines in 3D space. In other words, it is the quotient of two vectors.

Using the quaternion, the sensor data can be passed into other graphics libraries to display 3D objects and their orientation in 3D space.

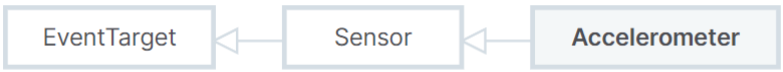
Upon error, all error messages will be logged to the console.

4.3 Accelerometer



The Accelerometer sensor reads the acceleration on the mobile device along all three axes. This sensor includes the force of gravity, unlike linear acceleration.

The Accelerometer object inherits from the Sensor base class:

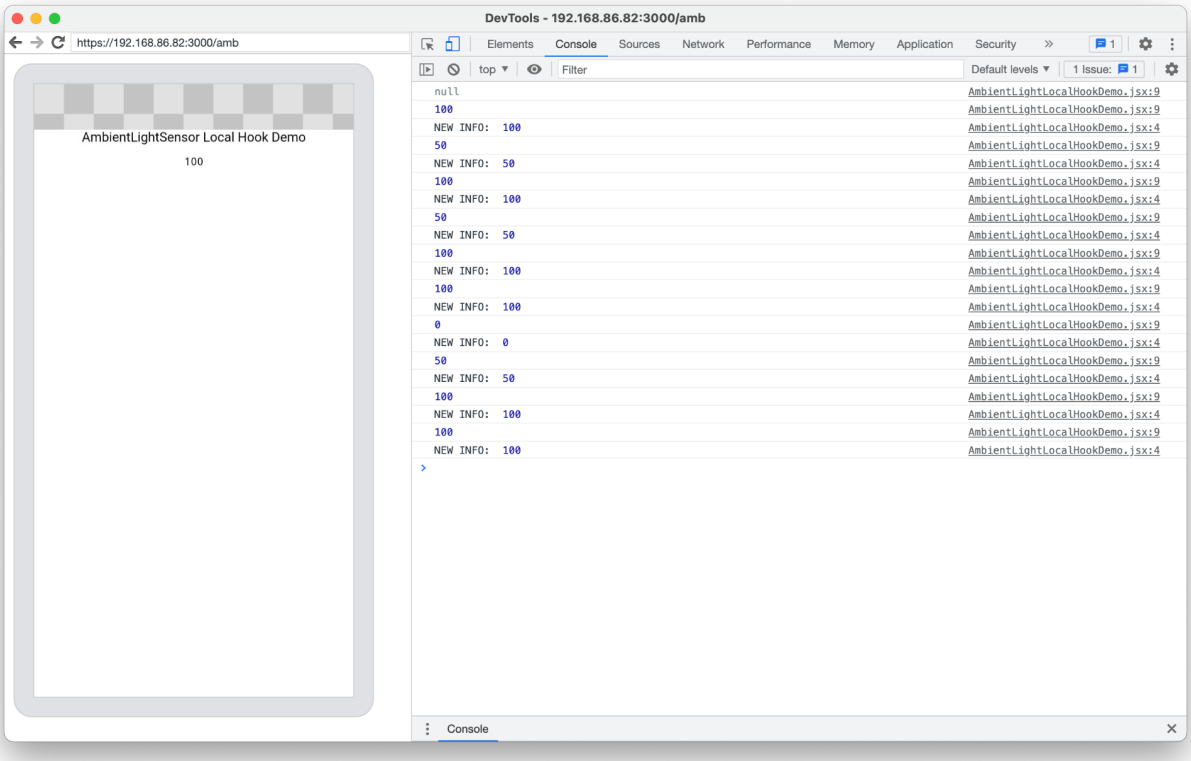


Source: https://developer.mozilla.org/en-US/docs/Web/API/Accelerometer

The React hook will read in the sensor data on each update, set the local state with the retrieved data.

Upon error, the error name and message will be written to the console log. Also, all three axes will be set to null in the state.

4.4 Ambient Light Sensor



The Ambient Light sensor detects light on the mobile device's main light detector. The units given in the data retrieved are in terms of lux units.

The Ambient Light sensor object inherits from the Sensor base class:

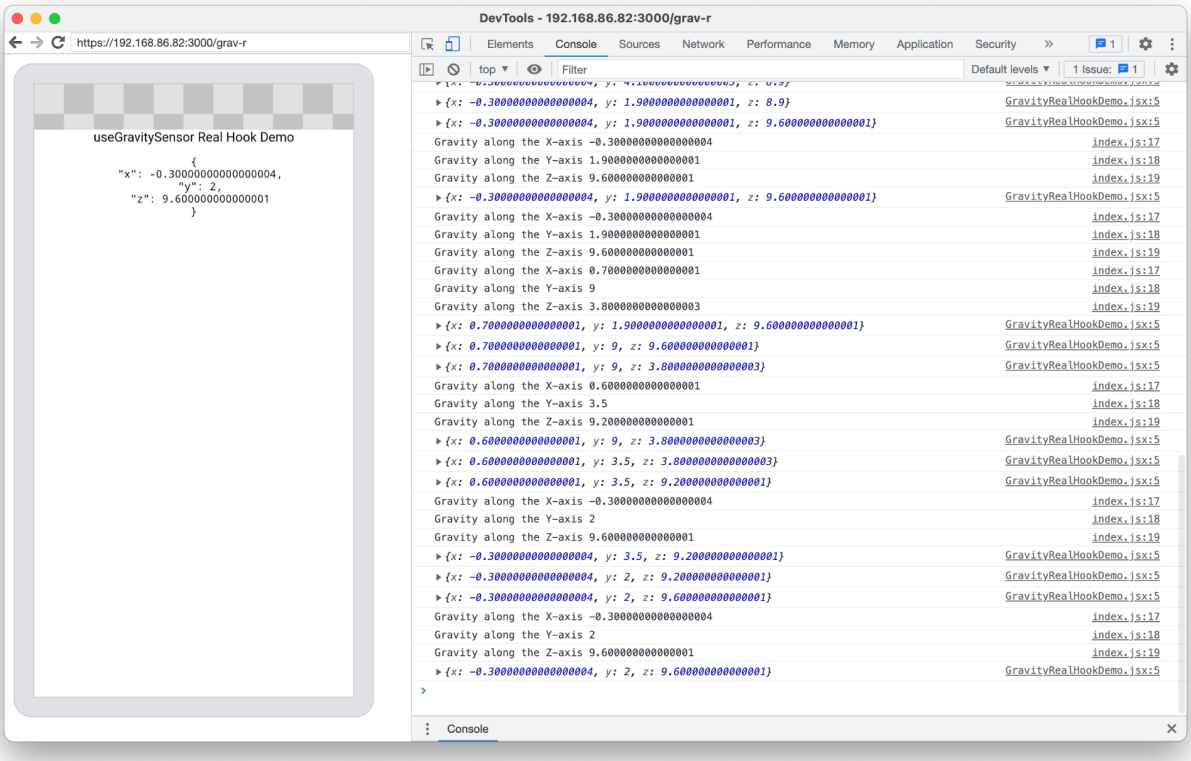
Source: https://developer.mozilla.org/en-US/docs/Web/API/AmbientLightSensor

The React hook will store the return lux unit data in state each time the Javascript built-in window object updates.

If a callback function is provided, that function will also be called on each update.

Upon error, the hook will print the error to the console log and set the current state to null.

4.5 Gravity



The gravity sensor interface provides information about the force of gravity applied to a mobile device along all three axes.

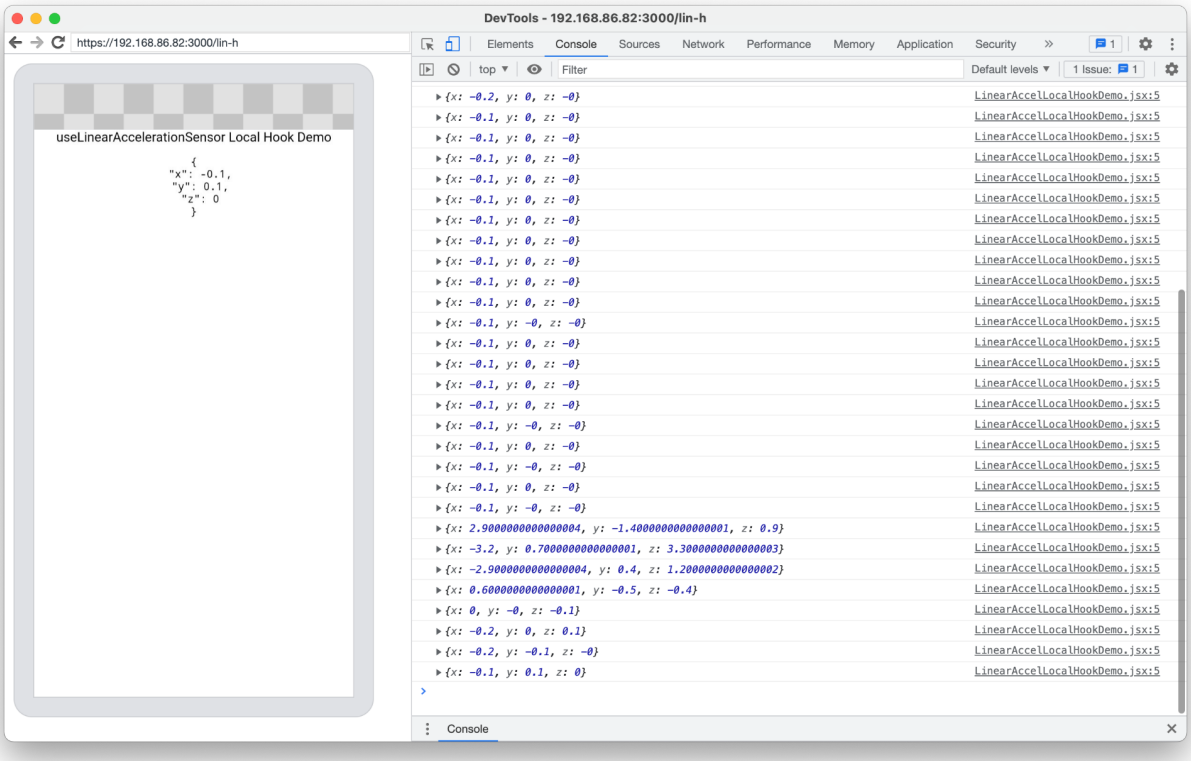
This sensor inherits properties from the Accelerometer sensor API.

Source: https://developer.mozilla.org/en-US/docs/Web/API/GravitySensor

The gravity sensor React hook stores each axis force in a separate state variable, and then returns all three as a Javascript object.

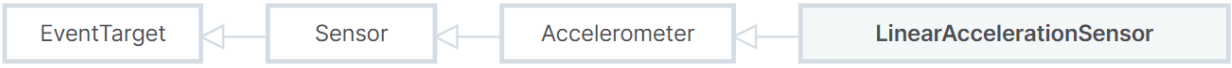
Upon error, then event error name and message are logged to the console.

4.6 Linear Acceleration



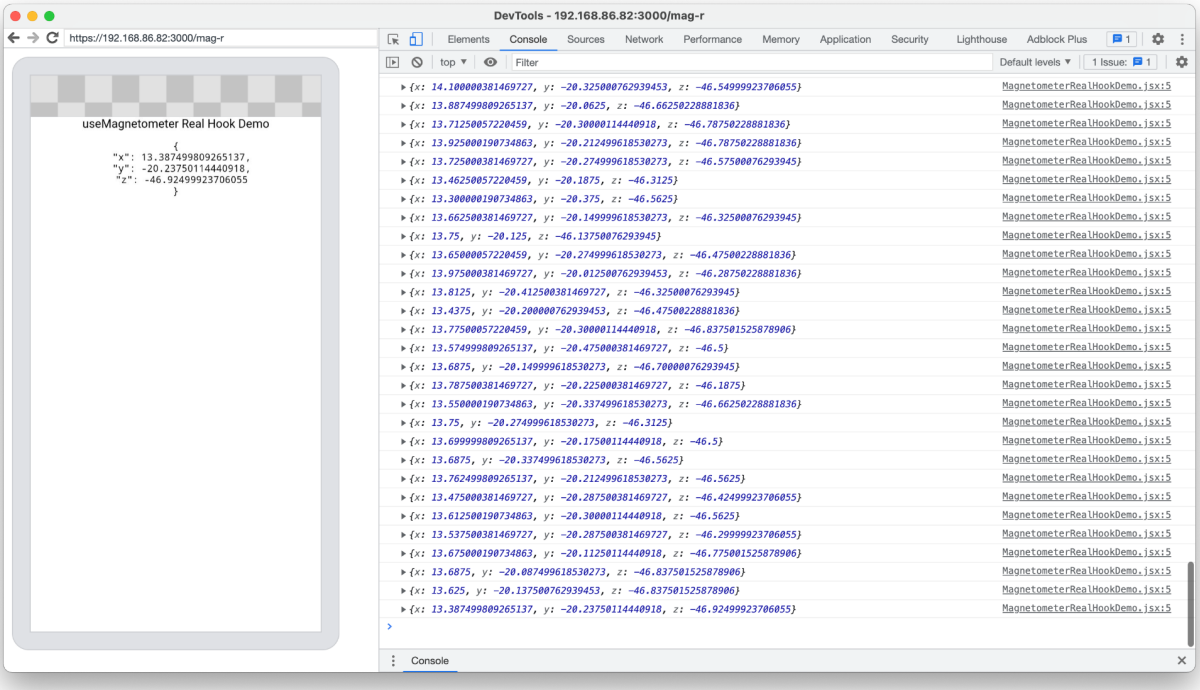
The Linear Acceleration Sensor reads acceleration on all three axes, but without taking into account the force of gravity.

This sensor inherits properties from its ancestors including the Accelerometer sensor as its immediate parent.



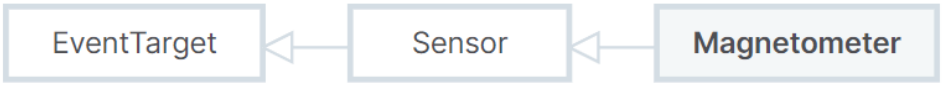
Source: https://developer.mozilla.org/en-US/docs/Web/API/LinearAccelerationSensor

4.7 Magnetometer



The Magnetometer sensor interface provides data about the magnetic field surrounding the mobile device's magnetometer sensor.

This sensor inherits only from the base Sensor interface:



Source: https://developer.mozilla.org/en-US/docs/Web/API/Magnetometer

The Magnetometer sensor React hook stores all three axis data points in an object in state. At a given frequency, the sensor will emit an event and then update the internal state of the React hook. The hook then sends the state data back to the caller.

Appendix A: Project Timeline

- 1/18/2022: Begin planning of project and determine backlog
- 1/25: Research motion sensors and the API docs on W3 and Mozilla
- 2/1: Research React and React Hooks and existing implementations
- 2/8: Complete Gyroscope React Hook
- 2/15: Review existing hook code and prepare for future hooks
- 2/22: Complete Absolute Orientation React Hook
- 3/1: Review existing code and refactor if necessary
- 3/8: Iterate on existing hooks with newly refactored code
- 3/15: Complete Accelerometer React Hook
- 3/22: Gather feedback on project, presentation, and methodologies
- 3/29: Complete Gravity Sensor React Hook
- 4/5: Review backlog and plan for future development
- 4/12: Complete Linear Acceleration React Hook
- 4/19: Prepare final codebase for deliver and publishing of all code
- 4/26: Review final demo and test all existing motion sensors