CopenhagenJS
September 2023

Building, testing, and releasing a multiplatform SDK using GitHub Actions

Kenneth Geisshirt
Lead Engineer @ MongoDB
https://github.com/kneth

Agenda

- About Realm JavaScript
- Continuous Integration and Delivery
- What is GitHub Actions?
- Our workflows in a nutshell
- Using caching to speed things up

About Realm JavaScript - 1

- Realm is an object database
  - Full ACID
  - Advanced query engine
- Synchronization with MongoDB
  - Object-Document Mapper
  - Eventually consistent

```
class Car extends Realm.Object {
  static schema = {
    name: "Car",
    properties: {
      _id: { type: 'objectId', default: () => new Realm.BSON.ObjectId() },
      make: "string",
      model: "string",
      miles: "int?",
    },
    primaryKey: '_id',
  };
}
```

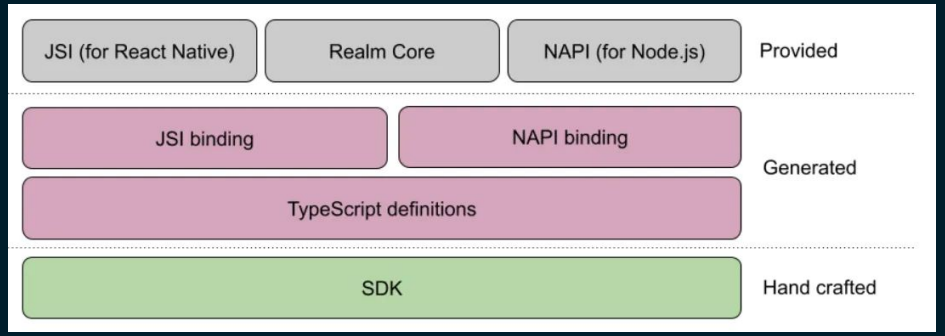
- Realm JavaScript is a JavaScript/TypeScript SDK for Realm
- Tight integration with JavaScript engines
  - V8 - node.js + Electron (Linux, Windows, MacOS)
  - JavaScriptCore and Hermes - React Native (iOS and Android)

```
let realm = await Realm.open({ schema: [Car] });
realm.write(() => {
  realm.create(Car, {
    make: “Opel”,
    model: “Astra”,
    miles: 10543 });
});
let opels = realm.objects(Car).filtered(“make == ‘Opel’”);
realm.close();
```

About Realm JavaScript - 2

- Realm Core is written in C++
- Integration with JS engines
  - JSI + NAPI
  - Generated C++ code
- Generate TypeScript definitions for Realm Core
- Public API (SDK) implemented in TypeScript

- The code generator is implemented in TypeScript



The architectural layers

No developer can manually

- Generate code for multiple JavaScript engines
- Build on Windows, MacOS and Linux on same machine
- Run 870+ tests on five different operating systems
- Lint TypeScript and C++ code constantly
- Upload binaries and API documentation when releasing

Automation is required

Continuous Integration and Delivery (CI/CD)

Automate everything

- Build project for all supported platform
- Lint your code source with predefined rules
- Spawn test servers
- Run tests on all supported platform
- Publish releases on NPM
- Notifications on Slack

Realm JavaScript

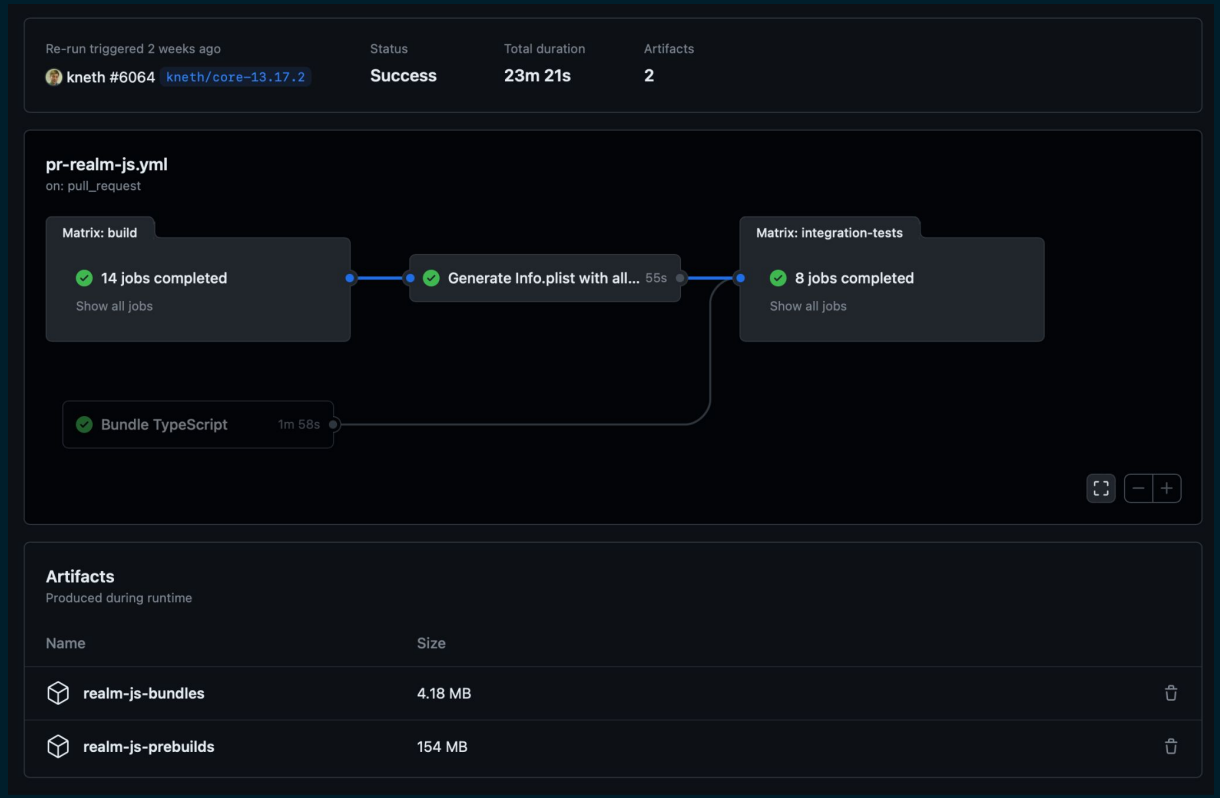
- Generate C++ and TypeScript binding using Code Generator
- Compile C++ code for five operating systems (gcc, clang, VSC++, xcode)
- Transpile TypeScript code
- Use ESLint and clang-format
- Cache artifacts to minimize build times
- Orchestrate test servers using Docker

What is GitHub Actions

- Automate workflows using YAML files
  - Workflow → jobs → steps
  - Build Matrix
- Predefined Github Runners
  - Linux, Windows, MacOS
  - Commonly used software installed: C++ compilers, node/npm
- Use 3rd party actions in your workflow
  - Checkout git repository
  - Select node version
- Workflow can be triggered by events
  - New issue or pull request created
  - Commits pushed to branch
  - Periodically (cron-like)
  - Started by a user

```yaml
name: Linting (Pull Request)
on: pull_request
concurrency:
 group: ${{ github.workflow }}-${{ github.ref }}
 cancel-in-progress: true
env:
 REALM_DISABLE_ANALYTICS: 1
jobs:
 lint:
   name: Lint
   runs-on: ubuntu-latest
   steps:
     - uses: actions/checkout@v3
       with:
         submodules: "recursive"
     - uses: actions/setup-node@v3
       with:
         node-version: 18
     - name: Install root package dependencies
       run: npm ci --ignore-scripts
     - name: Run linting of subpackages
       run: npm run lint
     - name: Run linting of C++ code
       run: npm run lint:cpp
```

Our workflow in a nutshell



21 workflows

- 1670 lines YAML code
- 14 build variants
- 8 test variants
- Install tests (daily against React Native)
- Releasing
  - 3 packages to npm
  - API docs to S3
- Janitor work
  - Auto-assign PRs
  - Issue labels
  - Clean up MongoDB clusters

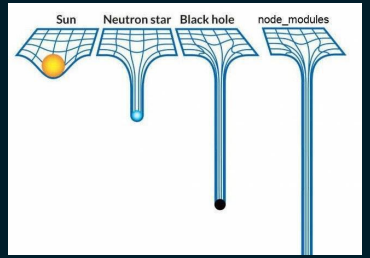
Using caching to speed things up

`Compiling ~240k lines C++ takes a while`

```
- name: ccache
  uses: hendrikmuhs/ccache-action@v1
  with:
    key: ${{ runner.os }}-${{ matrix.variant.os }}-${{
matrix.variant.arch }}
      max-size: '2.0G'

- name: Configure ccache
  run: ccache --set-config="compiler_check=content"

# Ignoring scripts to prevent a prebuilt from getting fetched
- name: Install dependencies
  run: npm ci --ignore-scripts

# build the c++ library for standard targets
- name: Build node
  if: ${{ (matrix.variant.os != 'ios') && (matrix.variant.os !=
'android') }}
    run: npm run build:node:prebuild:${{matrix.variant.arch}}
--workspace realm
```

`npm ci` downloads half of the internet

Or it feels like it




```
- name: Get NPM cache directory
  id: npm-cache-dir
  shell: bash
  run: echo "dir=$(npm config get cache)" >>
$GITHUB_OUTPUT

- name: Restore NPM cache
  id: npm-cache
  uses: actions/cache@v3
  with:
    path: ${{ steps.npm-cache-dir.outputs.dir }}
    key: ${{ runner.os }}-node-${{
hashFiles('package-lock.json') }}
    restore-keys: |
      ${{ runner.os }}-node-
```

Automation is king

Learn more

- Realm JavaScript
  - https://github.com/realm/realm-js
- Official documentation
  - https://docs.github.com/actions
- Collections of actions
  - https://github.com/marketplace?type=actions
- Learning Github Actions: Automation and Integration of Ci/Cd With Github
  - Published later this month