## What is Aurelia?

Aurelia is a collection of Modern JavaScript modules, which when used together, function as a powerful platform for building browser, desktop and mobile applications, all open source and built on open web standards.

Let's unpack this statement a little...

### A Collection of Modern JavaScript Modules

Rather than being a monolithic framework, Aurelia is broken down into a collection of feature-oriented modules. Examples of feature modules include metadata, dependency injection, binding, templating, router and more. Each module is written using ECMAScript (aka JavaScript) or TypeScript (a strict superset of JavaScript that adds compile-time type checking). Many of these modules can be used individually in any type of JavaScript project, including Node.js.

### A Powerful Platform for Building Apps

While Aurelia's modules can be used for many purposes, their true power lies in using them together as a front-end application platform. If your end goal is to create rich, engaging experiences for your customers, meeting or exceeding what is found in modern native applications, Aurelia provides the means. Through its rich component model, dynamic UI composition, routing and extensive set of plugins, Aurelia provides a comprehensive set of capabilities and tools to build any front-end experience you can dream up, whether you're targeting the browser, mobile or desktop.

### Open Source

With all the power and capability that Aurelia offers, you might expect it to have an expensive licensing model or be closed source, but neither is true. Aurelia is free and its code is open sourced under [the MIT License](http://opensource.org/licenses/MIT) , a very permissive license used by many popular web projects today. Its starter kits and documentation are available under [the Creative Commons 0](http://creativecommons.org/publicdomain/zero/1.0/legalcode) license. It also has [a Contributor License Agreement (CLA)](https://github.com/durandalproject/about/blob/master/CLA.md) for those who wish to join the core team in working on Aurelia. Ultimately, this means that you can use Aurelia without fear of legal repercussions and we can build it in the same confidence.

### Built on Open Web Standards

Not only is Aurelia written with ECMAScript, but it's also designed to make careful use of the DOM standard. Rather than Aurelia utilizing a costly abstraction over the DOM, implementing its own custom HTML parser or adopting framework-specific JavaScript extensions, it leverages the latest DOM APIs to get "bare metal" performance, exceptional memory efficiency and all while staying synced with the continuous improvements of the browser platform itself. Additionally, Aurelia's component model is based on W3C Web Components HTML Templates and ShadowDOM, so you know it will stand the test of time and will enable you to evolve your application along with advances in the standards, without major application re-writes or framework churn.

## Why choose Aurelia?

There are many frameworks to choose from today. We believe that Aurelia provides a fresh and exciting approach to front-end development with power and flexibility that is unmatched by other options. That said, we recognize that each team and each project has different needs. You might find Aurelia to be the right choice for you if...

* **You want an all-in-one solution** - Aurelia provides core capabilities like dependency injection, templating, routing and pub/sub, so you don't have to piece together a bunch of libraries in order to build an application. On top of this rich core, Aurelia also provides a number of additional plugins for internationalization, validation, modal dialogs, UI virtualization and much more. You also don't have to cobble together a bunch of different tools. Aurelia provides a CLI for generating and building projects, a browser plugin for debugging and a VS Code plugin as well. Yet, you're not forced to use any of these as Aurelia is structured to enable you to swap out any detail, even down to the templating/binding engine, in order to guarantee maximum flexibility.
* **You need blazing rendering speed and great memory efficiency** - In 3rd-party benchmarks like DB Monster, Aurelia renders faster than any other framework today. Because of its batched rendering and observable object pooling, Aurelia also utilizes less memory and causes less GC churn than other frameworks.
* **You require the safety of uni-directional data-flow, but need the productivity of data-binding** - Aurelia features an observable-based binding system that uses uni-directional data-flow by default, pushing data from your model into your view via a highly efficient, DOM-batching mechanism. Two-way binding can also be leveraged for HTML form controls, allowing for increased developer productivity, without sacrificing the safety of uni-directional flow or that of component encapsulation.
* **You desire API stability amidst a turbulent JavaScript landscape** - Aurelia follows Semver and works hard not to make breaking changes to its APIs. We're proud to say that we've continued to innovate and advance the platform while having no breaking changes to core framework APIs since our 1.0 release on July 27, 2016.
* **You value high standards compliance** - Focused on ES2015+ and W3C Web Components while avoiding unnecessary abstractions, Aurelia provides the cleanest and most standards-compliant component model you'll find anywhere.
* **You think a framework should "get out of your way"** - Aurelia is the only framework that lets you build components with plain, vanilla JavaScript/TypeScript. The framework stays out of your way so your code remains clean and easy to evolve over time.
* **You like programming models that are easy to learn and remember** - Because of its simple, consistent design, developers are able to learn a very small set of Aurelia patterns and APIs while unlocking limitless possibilities. Simple conventions help developers follow solid patterns and reduce the amount of code they have to write and maintain. This all results in less fiddling with the framework and more focus on the application.
* **You prefer a platform that integrates well with other frameworks and libraries** - Because of the extensible design of Aurelia and its strict adherence to web standards, it's easy to integrate Aurelia with any 3rd party library or framework, including jQuery, React, Polymer, Bootstrap, MaterializeCSS and many more.
* **You love or want to be a part of open source** - Aurelia is open sourced under the MIT license and doesn't add or remove special clauses or conditions to the license. We're proud of the work our community has done together and we'd love you to join in and help us make Aurelia better for everyone.
* **You thrive on being part of a welcoming community** - With one of the largest and most active developer gitter channels, a huge number of contributors and a large, active core team, Aurelia has an amazing community. Our core team and community love to welcome new developers and we all work hard to help each other succeed.

**Forward-thinking**

Written with next-generation EcmaScript. Integrates with Web Components. No external dependencies. Leverage the technology of the future but target today's mobile, desktop and browser environments.

**Modern Architecture**

Rather than taking the monolithic framework approach, Aurelia is composed of smaller, focused modules. Use them together as a full-featured framework or pick and choose to build a custom solution.

**Two-Way Databinding**

Our technology enables powerful two-way binding to any object. By using adaptive techniques we can select the most efficient way to observe each property in your model and automatically sync your UI with best-in-class performance.

**Extensible HTML**

Aurelia's extensible HTML compiler lets you create custom HTML elements, add custom attributes to existing elements and control template generation, all with full support for dynamic loading, databinding and high-performance batched rendering.

**Routing & UI Composition**

Leverage our advanced client-side router with its pluggable pipeline, dynamic route patterns, child routers and asynchronous screen activation. Don't need a router but need dynamic, data-driven UI composition? We do that too.

**MV\* with Conventions**

Who wants to waste time writing tons of configuration code for their MV\* architecture? Simply leverage conventions to make constructing your app effortless. Don't like the conventions? Plug in your own or drop them altogether.

**Broad Language Support**

Use ES5, ES 2015, ES 2016 and TypeScript. Aurelia's APIs were carefully designed to be consumed naturally from both today's and tomorrow's popular web programming languages.

**Testable**

By combining ES 2015 modules with a simple, yet powerful Dependency Injection Container, we make it easy for you to create highly cohesive, yet minimally coupled code, making unit testing a snap.

## Rendering the Root Component

The root component is set by calling aurelia.setRoot(). If no values are provided, this defaults to treating the element with the aurelia-app attribute as the DOM host for your app and app.js/app.html as the source for the root component. However, you can specify whatever you want, just like this:

export function configure(aurelia) {

aurelia.use

.standardConfiguration()

.developmentLogging();

aurelia.start().then(() => aurelia.setRoot('my-root', document.getElementById('some-element'));

}

This causes the my-root.js/my-root.html to be loaded as the root component and injected into the some-element HTML element.

## Bootstrapping Older Browsers

Aurelia was originally designed for Evergreen Browsers. This includes Chrome, Firefox, IE11 and Safari 8. However, we also support IE9 and above through the use of additional polyfills. To support these earlier browsers, you need the [requestAnimationFrame Polyfill](https://www.npmjs.com/package/raf" \t "_blank) and the[MutationObserver polyfill](https://github.com/megawac/MutationObserver.js) . Once you have installed these (via npm install --save-dev raf mutationobserver-shim), you'll need to adjust your code to load them before Aurelia is initialized.

## Making Resources Global

When you create a view in Aurelia, it is completely encapsulated. In the same way that you must import modules into an ES2015/TypeScript module, you must also import or require components into an Aurelia view. However, certain components are used so frequently across views that it can become very tedious to import them over and over again. To solve this problem, Aurelia lets you explicitly declare certain "view resources" as global. In fact, the configuration helper method defaultResources() mentioned above does just that. It takes the default set of view resources, such as repeat, if, compose, etc, and makes them globally usable in every view. You can do the same with your own components. Here's how we could make the my-component custom element, located in a resources subfolder of your project, globally available in all views.