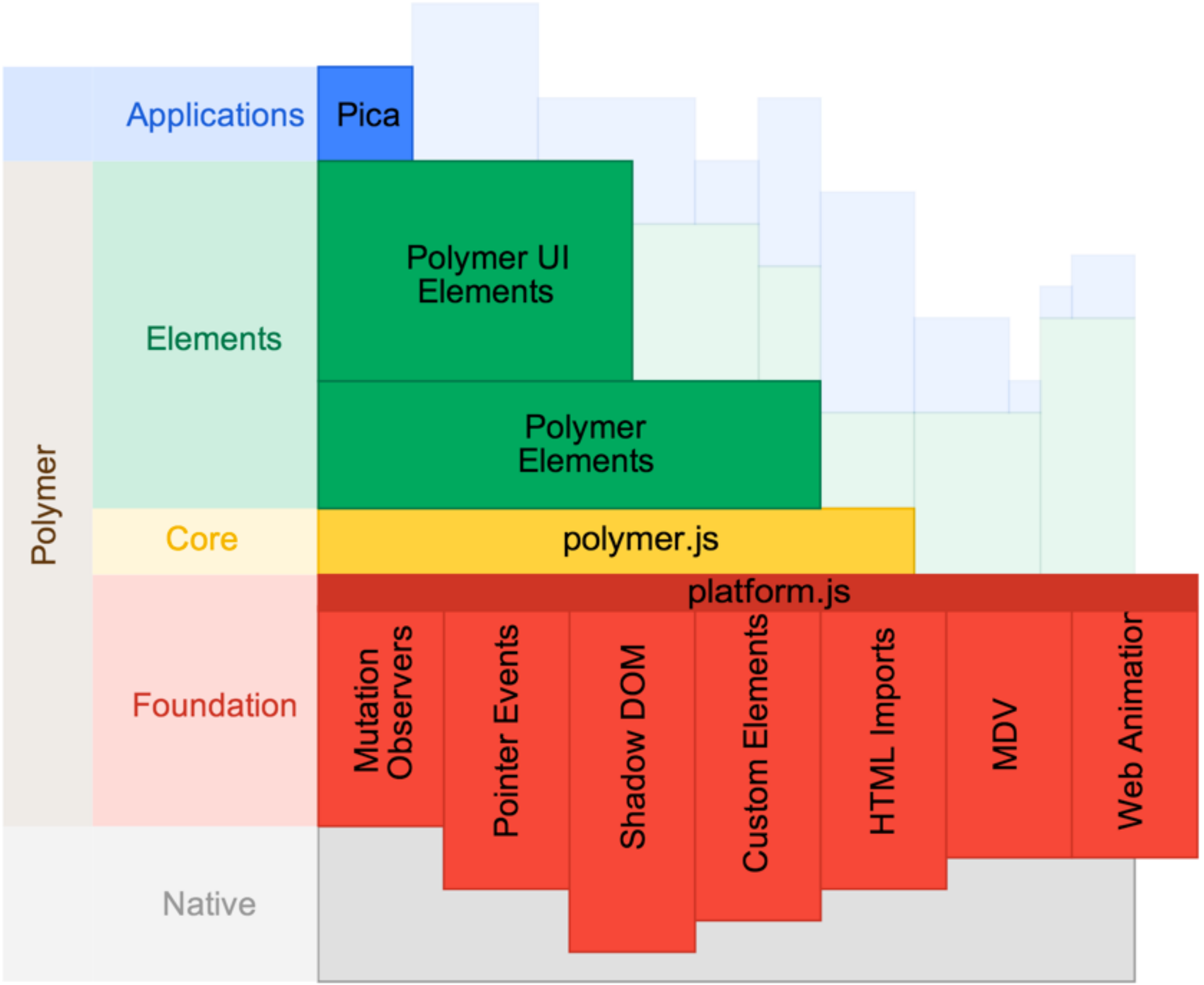
Why Polymer?

Fresno Summer Bootcamp 2015

Background

May 15, 2013


Applications
Pica
Elements
Polymer UI Elements
Polymer Elements
Core
polymer.js
Foundation
platform.js
Mutation Observers
Pointer Events
Shadow DOM
Custom Elements
HTML Imports
MDV
Web Animatior
Native
Polymer

June 26, 2014

Polymer core elements

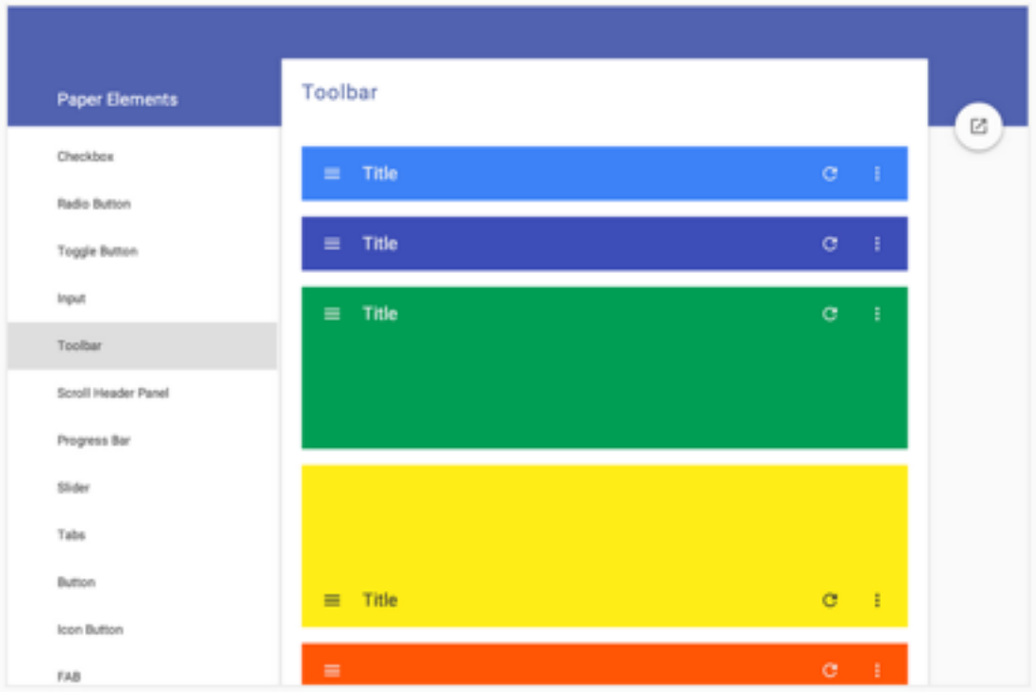
Polymer's core elements are a set of visual and non-visual utility elements. They include elements for working with layout, user input, selection, and scaffolding apps.

DOCS DEMOS

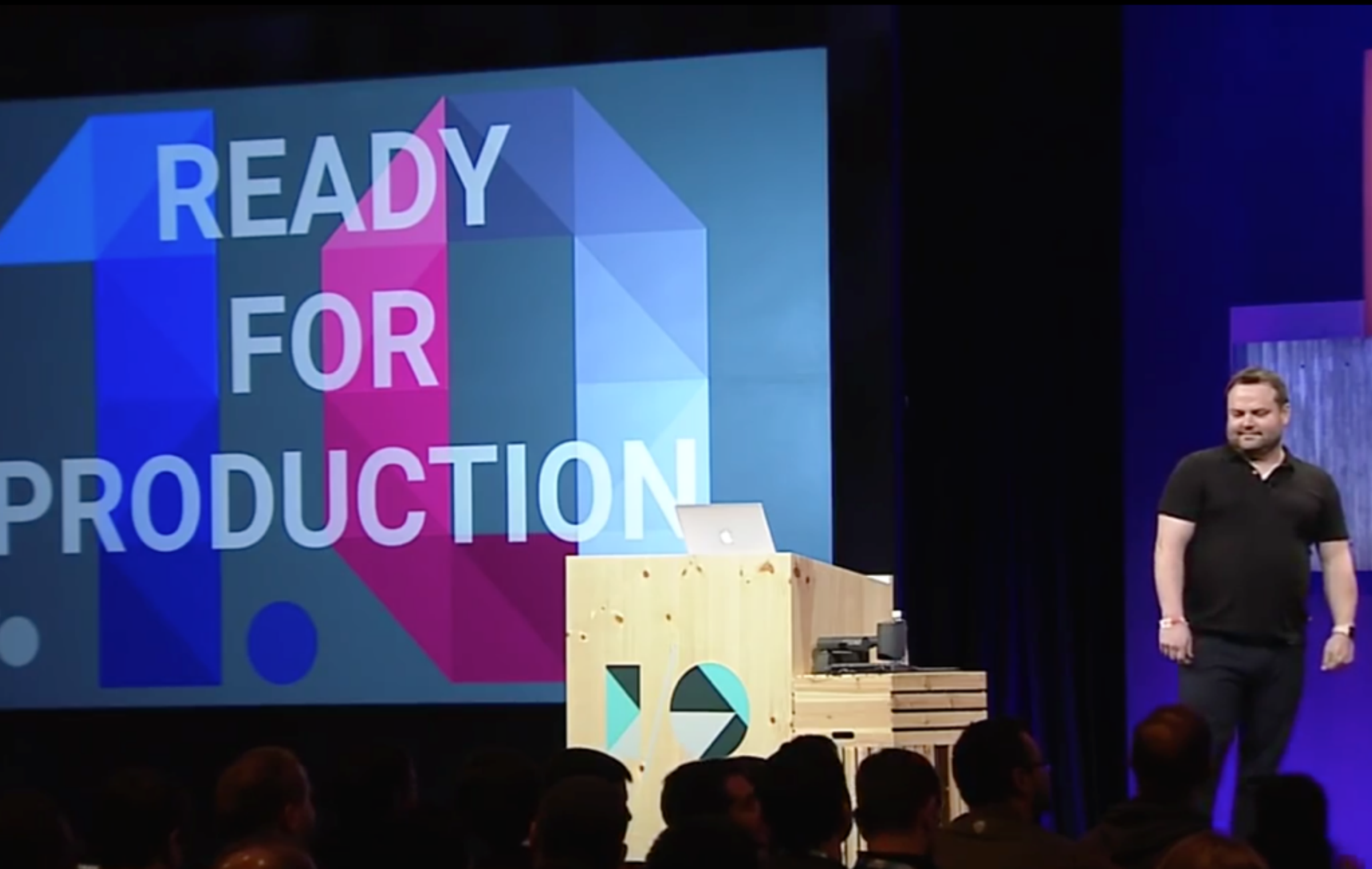
Paper elements

Polymer's paper elements collection implements material design for the web. They're a set of highly visual, highly interactive elements that include things like controls, layouts, hero transitions, and scrolling effects.

DOCS DEMOS



May 29, 2015



1.0.0

Fe

Iron Elements

Polymer core elements

1.0.1

Md

Paper Elements

Material design elements

1.0.1

Go

Google Web Components

Components for Google's APIs and services

1.0.1

Au

Gold Elements

Ecommerce Elements

1.0.0

Ne

Neon Elements

Animation and Special Effects

1.0.1

Pt

Platinum Elements

Offline, push, and more

1.0.0

Mo

Molecules

Wrappers for third-party libraries

Iron | **Paper** | ***...More*** — Element Collections

Polymer — Sugar Library

webcomponents.js — Optional Polyfills

Custom Elements | **HTML Templates** | **Shadow DOM** | **HTML Imports** — Browser APIs

Is Polymer a polyfill for Web Components?

What is a Polyfill?

- A polyfill, or polyfiller, is a piece of code (or plugin) that provides the technology that you, the developer, expect the browser to provide natively.

webcomponents.js

webcomponents.js

- A set of polyfills built on top of the Web Components specifications. It makes it possible for developers to use these standards today across all modern browsers.

webcomponents.js

- As these technologies are implemented in browsers, the polyfills will shrink and you'll gain the benefits of native implementations.

- *webcomponents.js* automatically detects native support and switches to the fast path when available. Your elements seamlessly start relying on the native stuff and get faster in the process.

webcomponents.js

- Although most developers will want to use everything in webcomponents.js, the polyfills are designed to be used separately, as well.

- A lighter *webcomponents-lite.js* build is included with the default download package including support for just Custom Elements and HTML Imports. This is useful if you don't require Shadow DOM, which is Polymer’s case.

Polyfill usage

Polyfill usage

```
$ bower install webcomponentsjs
```

Polyfill usage

```
<script src=“bower_components/webcomponentsjs/
webcomponents-lite.js"></script>
```

Polyfill usage

```
$ npm install webcomponents.js
```

Polyfill usage

```
<script src="node_modules/webcomponents.js/
webcomponents-lite.js"></script>
```

How is Polymer different from Angular, Ember, etc?

EMBER.JS GUIDES

Welcome to the Ember.js guides! This documentation will take you from total beginner to Ember expert. It is designed to start from the basics, and slowly increase to more sophisticated concepts until you know everything there is to know about building awesome web applications.

Most of these guides are designed to help you start building apps right away. If you'd like to know more about the thinking behind Ember.js, you'll find what you're looking for in the Understanding Ember.js section

These guides are written in Markdown and are available on GitHub. If there is something missing, or you find a typo or mistake, please help us by filing an issue or submitting a pull request. Thanks!

We're excited for all of the great apps you're going to build with Ember.js. To get started, select a topic from the left. They are presented in the order that we think will be most useful to you as you're learning Ember.js, but you can also jump to whatever seems most interesting.

Good luck!

We're done with Guides and Tutorials. Next up: Getting Started - Installing Ember

COMPREHENSIVE ROUTING

Design sophisticated views: map URL paths to application components, and use advanced features like nested and sibling routes. Angular 2 supports card stack navigation, animated transitions, and lazy loading for mobile users. If you already use routing from a prior version of Angular, you can easily migrate to Angular 2 routing.

ANIMATIONS

Tap directly into low-level animation support on mobile and desktop environments with easy-to-use Angular events. You can use CSS, JavaScript, and the Web Animations API to intelligently handle changes to animations in response to user events. Plan complex animation flows by sequencing the behavior of an entire website on a timeline.

HIERARCHICAL DEPENDENCY INJECTION

Angular 2 ships with powerful, yet simple-to-use dependency injection, allowing you to maintain modular applications without writing tedious glue code. Dependency injection helps you write tests by making it easy to inject test doubles.

What is the browser support for Polymer?

Chrome

- Chrome is the gold standard. Google has been leading to effort to spec and implement web components, and two of its features, shadow DOM and templates, are already shipping in Chrome's stable channel, while development is fast progressing on others.

Opera

- As Opera have switched to base their browsers on Chromium/Blink, I expect they will broadly follow Chrome in terms of implementation status.

Firefox

- Mozilla haven't been slouching either, and have made good progress towards implementing various parts of web components in Firefox.

Safari

- Although a number of web components features were implemented by Google within the WebKit codebase, they were never switched on in Safari. With the departure of the Chromium port, the WebKit maintainers have begun to remove features such as shadow DOM from the codebase. This is understandable from an engineering perspective, as unused code shouldn't be left lying around, but as a web developer it's disappointing to see.

Internet Explorer

- Microsoft have recently begun making their development plans for IE available at status.modern.ie. The four core web components features are, as yet, only **"under consideration"**. Therefore, it appears unlikely that any will ship as part of IE12. The most recent release, IE11, doesn't contain any of the web components APIs, but it does contain support for ancillary features such as Mutation Observers and Pointer Events that are useful for building components and polyfills.

Are We Componentized Yet?

Tracking the progress of Web Components through standardisation, polyfillification[1], and implementation.

	Specced	Implementation				
		Polyfill	Chrome / Opera	Firefox	Safari	IE
Templates			Stable	Stable	8	Vote
HTML Imports			Stable	On Hold		Vote
Custom Elements			Stable	Flag		Vote
Shadow DOM			Stable	Flag		Vote

Cells are clickable! Yellow means in-progress; green means mostly finished.

Last Update: ***6th February September, 2014*** *— Updated templates links and added IE voting links.*

Web Components!?

What are web components? They are encapsulated, reusable and composable widgets for the web platform. If that sounds cool, then I urge you to check out one of the very good introductions to them that others have provided. In my opinion, they're the most exciting thing to happen in web development since HTML5. Why? Because they promise to put the power and extensibility necessary to build sophisticated widgets and applications right into core web feature set. Imagine the capabilities of libraries like Angular, Ember, and

Obtaining Polymer

Obtaining Polymer

- If you’re ready to start your own project, you can install the Polymer library in one of several ways:
  - **Git**
  - **ZIP file**
  - **Bower**

Git

Polymer / polymer

Leverage the future of the web platform today.

2,874 commits **21** branches **61** releases **54** contributors

branch: **master** **polymer** / +

Clarify lifecycle details & limitations.

kevinpschaaf authored 2 days ago latest commit `19cdf42e5a`

explainer	Fixes #1241	4 months ago
src	Merge pull request #1855 from Polymer/1839-1854-kschaaf	3 days ago
test	Merge pull request #1855 from Polymer/1839-1854-kschaaf	3 days ago
.gitignore	Refactor build process	27 days ago
CHANGELOG.md	Add a CHANGELOG	4 days ago
CONTRIBUTING.md	CONTRIBUTING: fix blog link	22 days ago
LICENSE.txt	Add audit task	a month ago
PRIMER.md	Clarify lifecycle details & limitations.	2 days ago
README.md	Update README.md	6 days ago
bower.json	bump version for release	4 days ago
gulpfile.js	Use gulp-vulcanize concurrently for faster builds	13 days ago
index.html	Update docs. Add warnings.	27 days ago
package.json	bump version for release	4 days ago
polymer-micro.html	Add back Polymer.version string	26 days ago
polymer-mini.html	- actually iterate `behaviors` in reverse order for mixing in	a month ago
polymer.html	Merge pull request #1588 from Polymer/fix-inter-behavior-overrides	a month ago
wct.conf.json	Fix the config	2 months ago

Git

```
$ git clone https://github.com/Polymer/polymer.git
```

Git

- When you clone a component from GitHub, you need to manage all of the dependencies yourself.

- Because there are a number of dependencies, let’s try a different solution.

ZIP file

ZIP file

http://goo.gl/6u84fo

ZIP file

- Includes all dependencies, so you can unzip it and start using it immediately.

- The ZIP file requires no extra tools, but doesn’t provide a built-in method for updating dependencies.

Bower

Bower

- Bower removes the hassle of dependency management when developing or consuming elements. When you install a component, Bower makes sure any dependencies are installed as well.

- Bower also handles updating installed components.

Bower

A package manager for the web

Web sites are made of lots of things — frameworks, libraries, assets, utilities, and rainbows. Bower manages all these things for you.

Bower works by fetching and installing packages from all over, taking care of hunting, finding, downloading, and saving the stuff you're looking for. Bower keeps track of these packages in a manifest file, `bower.json`. How you use packages is up to you. Bower provides hooks to facilitate using packages in your tools and workflows.

Bower is optimized for the front-end. Bower uses a flat dependency tree, requiring only one version for each package, reducing page load to a minimum.

Install Bower

Bower is a command line utility. Install it with npm.

Bower

```
$ [sudo] npm install -g bower
```

Why Bower?

Olap Databasson
@BlueBoxTraveler

Follow

"What is Bower?"
"A package manager"
"How do I install it?"
"Use npm"
"What's npm?"
"A package manager"
"....."

RETWEETS 1,604 FAVORITES 665

9:06 PM - 8 Apr 2014

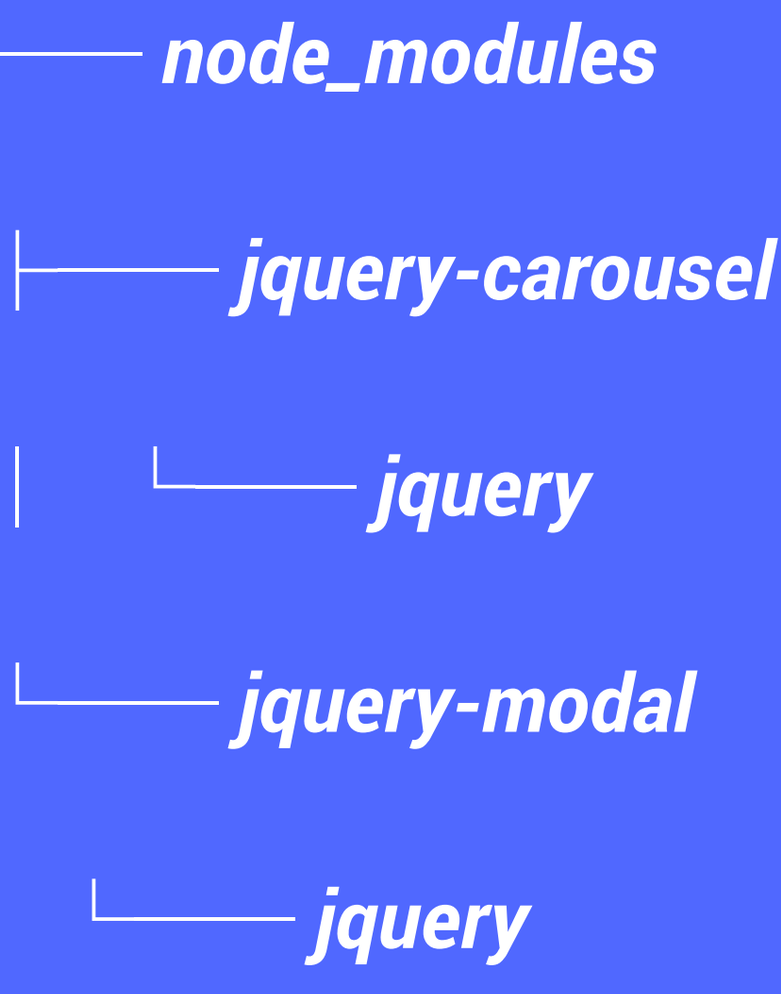
Why Bower?

Imagine you need these two packages. Both depend on a different version of the same package.

- *jquery-modal@2.0.0*
  - *jquery@1.0.0*
- *jquery-carousel@1.5.0*
  - *jquery@2.1.3*

Why Bower?

With **npm** you have a **nested** tree of dependencies.



Why Bower?

With **Bower** you have a **flat** tree of dependencies.

```
└──── bower_components
```

```
├──── jquery
```

```
├──── jquery-carousel
```

```
└──── jquery-modal
```

Bower usage

Bower usage

If you haven’t created a **bower.json** file for your application, run this command from the root of your project.

```
$ bower init
```

Bower usage

This generates a basic **bower.json** file. The next step is to install Polymer.

```
$ bower install Polymer/polymer#1.0.0
```

Bower usage

Include a **--save** to save the item as a dependency in your app's **bower.json**.

```
$ bower install --save Polymer/polymer#1.0.0
```

Bower usage

When a new version of Polymer is available, you can easily update your copy and its dependencies.

```
$ bower update
```

Semantic Versioning

Semantic Versioning

In the world of software management there exists a dread place called "dependency hell." The bigger your system grows and the more packages you integrate into your software, the more likely you are to find yourself, one day, in this pit of despair.

In systems with many dependencies, releasing new package versions can quickly become a nightmare. If the dependency specifications are too tight, you are in danger of version lock (the inability to upgrade a package without having to release new versions of every dependent package). If dependencies are specified too loosely, you will inevitably be bitten by version promiscuity (assuming compatibility with more future versions than is reasonable). Dependency hell is where you are when version lock and/or version promiscuity prevent you from easily and safely moving your project forward.

As a solution to this problem, I propose a simple set of rules and requirements that dictate how version numbers are assigned and incremented. For this system to work, you first need to declare a public API. This may consist of documentation or be enforced by the code itself. Regardless, it is important that this API be clear and precise. Once you identify your public API, you communicate changes to it with specific increments to your version number. Consider a version format of X.Y.Z (Major.Minor.Patch). Bug fixes not affecting the API increment the patch version, backwards compatible API additions/changes increment the minor version, and backwards incompatible API changes increment the major version.

I call this system "Semantic Versioning." Under this scheme, version numbers and the way they change convey meaning about the underlying code and what has been modified from one version to the next.

Semantic Versioning Specification (SemVer)

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Semantic Versioning

Given a version number **MAJOR.MINOR.PATCH**:

- **MAJOR** version when you make incompatible API changes,
- **MINOR** version when you add functionality in a backwards-compatible manner, and
- **PATCH** version when you make backwards-compatible bug fixes.

Semantic Versioning

- **>1.0.0** Include everything greater than 1.0.0
- **^1.0.0** Include everything greater than a particular version in the same major range
- **~1.0.0** Include everything greater than a particular version in the same minor range

npm semver calculator

New to semantic versioning? Learn the basics.

pick a package

lodash

enter a range

~1.0.0

0.4.0	0.6.0	0.9.1	1.0.1	2.0.0	3.0.0	3.5.0
0.4.1	0.6.1	0.9.2	1.1.0	2.1.0	3.0.1	3.6.0
0.4.2	0.7.0	0.10.0	1.1.1	2.2.0	3.1.0	1.0.2
0.5.0-rc.1	0.8.0	1.0.0-rc.1	1.2.0	2.2.1	3.2.0	3.7.0
0.5.0	0.8.1	1.0.0-rc.2	1.2.1	2.3.0	3.3.0	2.4.2
0.5.1	0.8.2	1.0.0-rc.3	1.3.0	2.4.0	3.3.1	3.8.0
0.5.2	0.9.0	1.0.0	1.3.1	2.4.1	3.4.0	3.9.0

how do i...

ything greater than a particular
e same major range

specify a range of stable versions

specify a range of prerelease version