Prism JS Plug-in

Prism JS

This is a generic topic that is designed to test sub-topics in a variety of locations. It should always be referenced from a topicref element; specialized elements in this test get their OWN, very special topics.

Prism is a lightweight, extensible syntax highlighter, built with modern web standards in mind. It's used in thousands of websites, including some of those you visit daily.

- **Dead simple** - Include prism.css and prism.js, use proper HTML5 code tags (code.language-xxxx), done!
- **Intuitive** - Language classes are inherited so you can only define the language once for multiple code snippets.
- **Light as a feather** - The core is 2KB minified & gzipped. Languages add 0.3-0.5KB each, themes are around 1KB.
- **Blazing fast** - Supports parallelism with Web Workers, if available.
- **Extensible** - Define new languages or extend existing ones. Add new features thanks to Prism's plugin architecture.
- **Easy styling** - All styling is done through CSS, with sensible class names like .comment, .string, .property etc

Contents

Chapter

1

prismjs

Topics:

Examples

The Prism source, highlighted with Prism (don't you just love how meta this is?):

```js
/* **********************************************
     Begin prism-core.js
********************************************** */

var _self = (typeof window !== 'undefined')
 ? window   // if in browser
 : (
   (typeof WorkerGlobalScope !== 'undefined' &&
 self instanceof WorkerGlobalScope)
  ? self // if in worker
  : {}   // if in node js
 );

/**
 * Prism: Lightweight, robust, elegant syntax highlighting
 * MIT license http://www.opensource.org/licenses/mit-license.php/
 * @author Lea Verou http://lea.verou.me
 */

var Prism = (function(){

// Private helper vars
var lang = /\blang(?:uage)?-([\w-]+)\b/i;
var uniqueId = 0;

var _ = _self.Prism = {
 manual: _self.Prism && _self.Prism.manual,
 disableWorkerMessageHandler: _self.Prism &&
 _self.Prism.disableWorkerMessageHandler,
 util: {
  encode: function (tokens) {
   if (tokens instanceof Token) {
    return new Token(tokens.type, _.util.encode(tokens.content),
 tokens.alias);
   } else if (_.util.type(tokens) === 'Array') {
    return tokens.map(_.util.encode);
   } else {
    return tokens.replace(/&/g, '&amp;').replace(/</g, '&lt;').replace(/
\u00a0/g, ' ');
   }
  },
 },

  type: function (o) {
   return Object.prototype.toString.call(o).match(/\[object (\w+)\]/)[1];
  },

  objId: function (obj) {
   if (!obj['__id']) {
    Object.defineProperty(obj, '__id', { value: ++uniqueId });
   }
   return obj['__id'];
  },

  // Deep clone a language definition (e.g. to extend it)
  clone: function (o, visited) {
```

```javascript
    var type = _.util.type(o);
    visited = visited || {};

    switch (type) {
     case 'Object':
      if (visited[_.util.objId(o)]) {
       return visited[_.util.objId(o)];
      }
      var clone = {};
      visited[_.util.objId(o)] = clone;

      for (var key in o) {
       if (o.hasOwnProperty(key)) {
        clone[key] = _.util.clone(o[key], visited);
       }
      }

      return clone;

     case 'Array':
      if (visited[_.util.objId(o)]) {
       return visited[_.util.objId(o)];
      }
      var clone = [];
      visited[_.util.objId(o)] = clone;

      o.forEach(function (v, i) {
       clone[i] = _.util.clone(v, visited);
      });

      return clone;
    }

    return o;
   }
  },

 };

 document.addEventListener('DOMContentLoaded', self.Prism.fileHighlight);

})();
```

This page's CSS code, highlighted with Prism:

```css
@import url(https://fonts.googleapis.com/css?family=Questrial);
@import url(https://fonts.googleapis.com/css?family=Arvo);

@font-face {
 src: url(https://lea.verou.me/logo.otf);
 font-family: 'LeaVerou';
}

/*
 Shared styles
 */

section h1,
#features li strong,
header h2,
```

```css
footer p {
 font: 100% Rockwell, Arvo, serif;
}

/*
 Styles
 */

* {
 margin: 0;
 padding: 0;
 font-weight: normal;
}

body {
 font: 100%/1.5 Questrial, sans-serif;
 tab-size: 4;
 hyphens: auto;
}

a {
 color: inherit;
}

section h1 {
 font-size: 250%;
}
```

This page's HTML, highlighted with Prism:

```html
<!DOCTYPE html>
<html lang="en">
<head>

<script>
 // Just a lil' script to show off that inline JS gets highlighted
 window.console && console.log('foo');
</script>
<meta charset="utf-8" />
<link rel="icon" href="favicon.png" />
<title>Prism</title>
<link rel="stylesheet" href="style.css" />
<link rel="stylesheet" href="themes/prism.css" data-noprefix />
<script src="prefixfree.min.js"></script>

<script>var
 _gaq = [['_setAccount', 'UA-33746269-1'], ['_trackPageview']];</script>
<script src="https://www.google-analytics.com/ga.js" async></script>
</head>
<body>

<header>
 <div class="intro" data-src="templates/header-main.html" data-type="text/
html"></div>

 <ul id="features">
  <li>
   <strong>Dead simple</strong>
```

```
    Include prism.css and prism.js, use proper HTML5 code tags
 (<code>code.language-xxxx</code>), done!
  </li>
  <li>
   <strong>Intuitive</strong>
   Language classes are inherited so you can only define the language once
 for multiple code snippets.
  </li>
  <li>
   <strong>Light as a feather</strong>
   The core is 2KB minified &amp; gzipped. Languages add 0.3-0.5KB each,
 themes are around 1KB.
  </li>
  <li>
   <strong>Blazing fast</strong>
   Supports parallelism with Web Workers, if available.
  </li>
  <li>
   <strong>Extensible</strong>
   Define new languages or extend existing ones.
   Add new features thanks to Prism's plugin architecture.
  </li>
  <li>
   <strong>Easy styling</strong>
   All styling is done through CSS, with sensible class names
 like <code>.comment</code>, <code>.string</code>, <code>.property</code>
 etc
  </li>
  </ul>

</header>

<section id="used-by">
 <h1>Used By</h1>

 <p>Prism is used on several websites, small and large. Some of them
 are:</p>

 <div class="used-by-logos">
  <a href="https://www.smashingmagazine.com/" target="_blank"><img src="img/
logo-smashing.png" alt="Smashing Magazine" /></a>
  <a href="http://alistapart.com/" target="_blank"><img src="img/logo-
ala.png" alt="A List Apart" /></a>
  <a href="https://developer.mozilla.org/" target="_blank"><img src="img/
logo-mdn.png" alt="Mozilla Developer Network (MDN)" /></a>
  <a href="https://css-tricks.com/" target="_blank"><img src="img/logo-css-
tricks.png" alt="CSS-Tricks" /></a>
  <a href="https://www.sitepoint.com/" target="_blank"><img src="img/logo-
sitepoint.png" alt="SitePoint" /></a>
  <a href="https://www.drupal.org/" target="_blank"><img src="img/logo-
drupal.png" alt="Drupal" /></a>
  <a href="https://reactjs.org/" target="_blank"><img src="img/logo-
react.png" alt="React" /></a>
  <a href="https://stripe.com/" target="_blank"><img src="img/logo-
stripe.png" alt="Stripe" /></a>
 </div>
</section>

<section id="examples">
 <h1>Examples</h1>

 <p>The Prism source, highlighted with Prism (don't you just love how meta
 this is?):</p>
 <pre data-src="prism.js"></pre>
```

```html
<p>This page's CSS code, highlighted with Prism:</p>
<pre data-src="style.css"></pre>

<p>This page's HTML, highlighted with Prism:</p>
<pre data-src="index.html"></pre>

<p>This page's logo (SVG), highlighted with Prism:</p>
<pre data-src="logo.svg"></pre>

<p>If you're still not sold, you can <a href="examples.html">view more
examples</a> or <a href="test.html">try it out for yourself</a>.</p>
</section>

<section id="features-full" class="language-markup">
 <h1>Full list of features</h1>
 <ul>
  <li><strong>Only 2KB</strong> minified &amp; gzipped (core). Each language
definition adds roughly 300-500 bytes.</li>
  <li>Encourages good author practices. Other highlighters encourage or even
force you to use elements that are semantically wrong,
   like <code>&lt;pre></code> (on its own) or <code>&lt;script></code>.
   Prism forces you to use the correct element for marking up
code: <code>&lt;code></code>.
   On its own for inline code, or inside a <code>&lt;pre></code> for blocks
of code.
   In addition, the language is defined through the way recommended in the
HTML5 draft: through a <code>language-xxxx</code> class.</li>
  <li>The language definition is inherited. This means that if multiple code
snippets have the same language, you can just define it once, in one of
their common ancestors.</li>
  <li>Supports <strong>parallelism with Web Workers</strong>, if available.
 Disabled by default (<a href="faq.html#why-is-asynchronous-highlighting-
disabled-by-default">why?</a>).</li>
  <li>Very easy to extend without modifying the code, due to
 Prism's <a href="#plugins">plugin architecture</a>. Multiple hooks are
scattered throughout the source.</li>
  <li>Very easy to <a href="extending.html#language-definitions">define
 new languages</a>. Only thing you need is a good understanding of regular
expressions</li>
  <li>All styling is done through CSS, with <a href="faq.html#how-do-i-know-
which-tokens-i-can-style-for">sensible class names</a> rather than ugly
namespaced abbreviated nonsense.</li>
  <li>Wide browser support: IE9+, Firefox, Chrome,
 Safari, <a href="faq.html#this-page-doesnt-work-in-opera">Opera</a>, most
Mobile browsers</li>
  <li>Highlights embedded languages (e.g. CSS inside HTML, JavaScript inside
HTML)</li>
  <li>Highlights inline code as well, not just code blocks</li>
  <li>Highlights nested languages (CSS in HTML, JavaScript in HTML)</li>
  <li>It doesn't force you to use any Prism-specific markup, not even a
 Prism-specific class name, only standard markup you should be using anyway.
 So, you can just try it for a while, remove it if you don't like it and
leave no traces behind.</li>
  <li>Highlight specific lines and/or line ranges
 (requires <a href="plugins/line-highlight/">plugin</a>)</li>
  <li>Show invisible characters like tabs, line breaks etc
 (requires <a href="plugins/show-invisibles/">plugin</a>)</li>
  <li>Autolink URLs and emails, use Markdown links in comments
 (requires <a href="plugins/autolinker/">plugin</a>)</li>
 </ul>
</section>

<section id="limitations">
```

```html
<h1>Limitations</h1>
<ul>
  <li>Any pre-existing HTML in the code will be stripped
off. <a href="faq.html#if-pre-existing-html-is-stripped-off-how-can-i-
highlight">There are ways around it though</a>.</li>
  <li>Regex-based so it *will* fail on certain edge cases, which are
documented in the <a href="examples.html">Examples section</a>.</li>
  <li>No IE 6-8 support. If someone can read code, they are probably in the
85% of the population with a modern browser.</li>
</ul>
</section>

<section id="basic-usage" class="language-markup">
<h1>Basic usage</h1>

<p>You will need to include the <code>prism.css</code>
and <code>prism.js</code> files you downloaded in your page. Example:
<pre><code>&lt;!DOCTYPE html>
&lt;html>
&lt;head>
 ...</code>
<code class="highlight">&lt;link href="themes/prism.css" rel="stylesheet" /
></code>
<code>&lt;/head>
&lt;body>
 ...</code>
<code class="highlight">&lt;script src="prism.js">&lt;/script></code>
<code>&lt;/body>
&lt;/html></code></pre>

<p>Prism does its best to encourage good authoring practices. Therefore,
it only works with <code>&lt;code></code> elements, since marking up code
without a <code>&lt;code></code> element is semantically invalid.
<a href="https://www.w3.org/TR/html52/textlevel-semantics.html#the-code-
element">According to the HTML5 spec</a>, the recommended way to define a
code language is a <code>language-xxxx</code> class, which is what Prism
uses.
Alternatively, Prism also supports a shorter version: <code>lang-
xxxx</code>.</p>
<p>To make things easier however, Prism assumes that this language
definition is inherited. Therefore, if multiple <code>&lt;code></code>
elements have the same language, you can add the <code>language-xxxx</code>
class on one of their common ancestors.
 This way, you can also define a document-wide default language, by
adding a <code>language-xxxx</code> class on the <code>&lt;body></code>
or <code>&lt;html></code> element.</p>

<p>If you want to opt-out of highlighting for a <code>&lt;code></code>
element that is a descendant of an element with a declared code language,
you can add the class <code>language-none</code> to it (or any non-existing
language, really).</p>

<p>The <a href="https://www.w3.org/TR/html5/grouping-content.html#the-pre-
element">recommended way to mark up a code block</a>
 (both for semantics and for Prism) is a <code>&lt;pre></code> element with
a <code>&lt;code></code> element inside, like so:</p>
<pre><code>&lt;pre>&lt;code class="language-css">p { color: red }&lt;/
code>&lt;/pre></code></pre>
<p>If you use that pattern, the <code>&lt;pre></code> will automatically
get the <code>language-xxxx</code> class (if it doesn't already have it)
and will be styled as a code block.</p>

<p>If you want to prevent any elements from being automatically
highlighted, you can use the attribute <code>data-manual</code>
```

```
 on the <code>&lt;script></code> element you used for prism and use
 the <a href="extending.html#api">API</a>.
 Example:</p>
<pre><code>&lt;script src="prism.js" data-manual>&lt;/script></code></pre>

 <h2>Usage with Webpack, Browserify, & Other Bundlers</h2>

 <p>If you want to use Prism with a bundler, install Prism
 with <code>npm</code>:</p>

 <pre><code>$ npm install prismjs</code></pre>

 <p>You can then <code class="language-js">import</code> into your
 bundle:</p>

 <pre><code class="language-js">import Prism from 'prismjs';</code></pre>

 <p>To make it easy to configure your Prism instance with only the languages
 and plugins you need, use the babel plugin,
  <a href="https://github.com/mAAdhaTTah/babel-plugin-prismjs">babel-plugin-
prismjs</a>. This will allow you to load
  the minimum number of languages and plugins to satisfy your needs.
  See that plugin's documentation for configuration details.</p>

 <h2>Usage with Node</h2>

 <p>If you want to use Prism on the server or through the command line,
 Prism can be used with Node.js as well.
 This might be useful if you're trying to generate static HTML pages with
 highlighted code for environments that don't support browser-side JS,
 like <a href="https://www.ampproject.org/">AMP pages</a>.</p>

 <p>Example:</p>
 <pre><code class="language-js">var Prism = require('prismjs');

// The code snippet you want to highlight, as a string
var code = "var data = 1;";

// Returns a highlighted HTML string
var html = Prism.highlight(code, Prism.languages.javascript,
 'javascript');</code></pre>

 <p>Requiring <code>prismjs</code> will load the default
 languages: <code>markup</code>, <code>css</code>,
  <code>clike</code> and <code>javascript</code>. You can load more
 languages with the
  <code class="language-javascript">loadLanguages()</code> utility, which
 will automatically handle any required dependencies.</p>
 <p>Example:</p>

 <pre><code class="language-js">var Prism = require('prismjs');
var loadLanguages = require('prismjs/components/');
loadLanguages(['haml']);

// The code snippet you want to highlight, as a string
var code = "= ['hi', 'there', 'reader!'].join \" \"";

// Returns a highlighted HTML string
var html = Prism.highlight(code, Prism.languages.haml, 'haml');</code></pre>

 <p><strong>Note</strong>: Do <em>not</em> use <code class="language-
javascript">loadLanguages()</code> with Webpack or another bundler, as this
 will cause Webpack to include all languages and plugins. Use the babel
 plugin described above.</p>
```

```
</section>

<section id="languages-list" class="language-markup">
 <h1>Supported languages</h1>
 <p>This is the list of all <span id="languages-list-count"></span>
 languages currently supported by Prism, with
  their corresponding alias, to use in place of <code>xxxx</code> in
 the <code>language-xxxx</code> (or <code>lang-xxxx</code>) class:</p>
</section>

<section id="plugins">
 <h1>Plugins</h1>
 <p>Plugins are additional scripts (and CSS code) that extend Prism's
 functionality. Many of the following plugins are official, but are released
 as plugins to keep the Prism Core small for those who don't need the extra
 functionality.</p>
 <ul class="plugin-list"></ul>

 <p>No assembly required to use them. Just select them in
 the <a href="download.html">download</a> page.</p>
 <p>It's very easy to <a href="extending.html#writing-plugins">write
 your own Prism plugins</a>. Did you write a plugin for Prism that
 you want added to this list? <a href="https://github.com/LeaVerou/
prism" target="_blank">Send a pull request</a>!</p>
</section>

<section id="languages">
 <h1>Third-party language definitions</h1>

 <ul>
  <li><a href="https://github.com/SassDoc/prism-scss-sassdoc">SassDoc Sass/
Scss comments</a></li>
 </ul>
</section>

<section id="tutorials">
 <h1>Third-party tutorials</h1>

 <p>Several tutorials have been written by members of the community
 to help you integrate Prism into multiple different website types and
 configurations:</p>

 <ul>
  <li><a href="https://websitebeaver.com/escape-html-inside-code-or-
pre-tag-to-entities-to-display-raw-code-with-prismjs">Escape HTML
 Inside &lt;code&gt; or &lt;pre&gt; Tag to Entities to Display Raw Code with
 PrismJS</a></li>
  <li><a href="http://crambler.com/how-to-implement-prism-js-syntax-
highlighting-into-your-wordpress-site/">How To Implement Prism.js Syntax
 Highlighting Into Your WordPress Site</a></li>
  <li><a href="http://wp.tutsplus.com/tutorials/plugins/adding-a-syntax-
highlighter-shortcode-using-prism-js/">Adding a Syntax Highlighter Shortcode
 Using Prism.js | WPTuts+</a></li>
  <li><a href="https://www.stramaxon.com/2012/07/prism-syntax-highlighter-
for-blogger.html">Implement PrismJs Syntax Highlighting to your Blogger/
BlogSpot</a></li>
  <li><a href="http://www.allblogtools.com/tricks-and-hacks/beautify-source-
codes-in-your-posts-with-prism-syntax-highlighter-for-blogger/">Beautify
 Source Codes In Your Posts With Prism Syntax Highlighter For
 Blogger</a></li>
  <li><a href="https://schier.co/blog/2013/01/07/how-to-re-run-prismjs-on-
ajax-content.html">How To Re-Run Prism.js On AJAX Content</a></li>
```

```html
  <li><a href="https://www.semisedlak.com/highlight-your-code-syntax-with-
prismjs">Highlight your code syntax with Prism.js</a></li>
  <li><a href="https://usetypo3.com/fs-code-snippet.html">A code snippet
 content element powered by Prism.js for TYPO3 CMS</a></li>
  <li><a href="https://auralinna.blog/post/2017/code-syntax-highlighting-
with-angular-and-prismjs">Code syntax highlighting with Angular and
 Prism.js</a></li>
 </ul>

 <p>Please note that the tutorials listed here are not verified to contain
 correct information. Read at your risk and always check the official
 documentation here if something doesn't work :)</p>

 <p>Have you written a tutorial about Prism that's not already included
 here? Send a pull request!</p>
</section>

<section id="credits">
 <h1>Credits</h1>
 <ul>
  <li>Special thanks to <a href="https://github.com/Golmote">Golmote</a>
 and <a href="https://github.com/apfelbox">Jannik Zschiesche</a> for their
 contributions and for being <strong>amazing maintainers</strong>. Prism
 would not have been able to keep up without their help.</li>
  <li>To <a href="https://twitter.com/kizmarh">Roman Komarov</a> for his
 contributions, feedback and testing.</li>
  <li>To <a href="https://twitter.com/zdfs">Zachary Forrest</a>
 for <a href="https://twitter.com/zdfs/statuses/217834980871639041">coming
 up with the name "Prism"</a></li>
  <li>To <a href="https://stellarr.deviantart.com/">stellarr</a> for
 the <a href="https://stellarr.deviantart.com/art/Spectra-Wallpaper-
Pack-97785901">spectrum background</a> used on this page</li>
  <li>To <a href="https://twitter.com/thecodezombie">Jason
 Hobbs</a> for <a href="https://twitter.com/thecodezombie/
status/217663703825399809">encouraging me</a> to release this script as
 standalone</li>
 </ul>
</section>

<footer data-src="templates/footer.html" data-type="text/html"></footer>

<script src="prism.js"></script>
<script src="utopia.js"></script>
<script src="components.js"></script>
<script src="code.js"></script>
<script>
(function() {
 var languageItems = [];
 var languages = components.languages;
 var count = 0;
 for (var id in languages) {
  if (id == 'meta') {
   continue;
  }
  count++;
  var name = languages[id].title || languages[id];

  languageItems.push({
   tag: 'li',
   attributes: {
    'data-id': id
   },
   contents: [
    name,
```

```
    ' - ',
    {
     tag: 'code',
     contents: id
    }
   ]
  });
 }
 $u.element.create('ul', {
  contents: languageItems,
  inside: '#languages-list'
 });
 $u.element.contents($('#languages-list-count'),
 count);
}());
</script>

</body>
</html>
```

This page's logo (SVG), highlighted with Prism:

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 200 170">
<path fill="#fff" d="M55.37 131.5H48.4v9.13h6.97c1.67 0 2.92-.4
3.78-1.22.85
-.8 1.28-1.92 1.28-3.33s-.43-2.54-1.28-3.35c-.86-.8-2.12-1.2-3.78-1.2m29.52
6.4c.3-.53.47-1.2.47-2.04 0-1.35-.45-2.4-1.37-3.2-.92-.76-2.14-1.15-3.65
-1.15H72.9v8.52h7.32c2.26 0 3.82-.7 4.67-2.1M100 0L0 170h200L100 0M60.86
141.03c-1.3 1.22-3.1 1.84-5.33 1.84H48.4v7.55H46v-21.2h9.53c2.24 0 4.02.63
5.34 1.87 1.3 1.23 1.96 2.88 1.96 4.95 0 2.1-.66 3.75-1.97 4.98m24.5 9.4l
-5.1-8.14h-7.37v8.12h-2.4v-21.2h10.14c2.15 0 3.88.6 5.18 1.8 1.3 1.18 1.95
2.8 1.95 4.84 0 2.64-1.1 4.44-3.3 5.4-.6.28-1.22.5-1.82.6l5.57 8.56h-2.85m
13.43 0h-2.4v-21.2h2.4v21.2m23.56-1.32c-1.48 1.05-3.53 1.57-6.16 1.57-2.96
0
-5.23-.6-6.78-1.85-1.4-1.1-2.18-2.7-2.37-4.74h2.5c.08 1.45.78 2.56 2.1 3.33
1.16.67 2.68 1 4.58 1 3.97 0 5.95-1.25 5.95-3.74
0-.86-.35-1.53-1.07-2.02-.7
-.5-1.6-.9-2.68-1.2-1.07-.33-2.24-.63-3.48-.9s-2.4-.65-3.5-1.08-1.97-1.02
-2.68-1.73c-.7-.72-1.07-1.68-1.07-2.9 0-1.73.65-3.13 1.97-4.22 1.32-1.08
3.32-1.62 6-1.62 2.67 0 4.75.6 6.23 1.85 1.34 1.1 2.05 2.5 2.14 4.2h-2.46c
-.22-1.76-1.35-2.92-3.4-3.5-.72-.2-1.62-.3-2.7-.3s-1.98.1-2.72.35c-.74.25
-1.3.55-1.7.9-.42.35-.7.74-.83 1.17s-.2.88-.2 1.36c0 .5.2.93.62 1.33s.96.75
1.65 1.03c.68.28 1.46.52 2.33.73.88.2 1.77.43 2.67.65.9.22 1.8.48
2.68.77.87
.3 1.65.65 2.33 1.1 1.53.96 2.28 2.27 2.28 3.94 0 2-.74 3.5-2.22 4.55m28.84
1.32v-17.54l-7.84 10.08-7.97-10.08v17.54H133v-21.2h2.78l7.58 10.06 7.45
-10.05h2.8v21.2h-2.4"/>
</svg>
```

If you're still not sold, you can view more examples or try it out for yourself.

Full list of features

- **Only 2KB** minified & gzipped (core). Each language definition adds roughly 300-500 bytes.
- Encourages good author practices. Other highlighters encourage or even force you to use elements that are semantically wrong, like `<pre>` (on its own) or `<script>`. Prism forces you to use the correct element for

marking up code: `<code>`. On its own for inline code, or inside a `<pre>` for blocks of code. In addition, the language is defined through the way recommended in the HTML5 draft: through a `language-xxxx` class.

- The language definition is inherited. This means that if multiple code snippets have the same language, you can just define it once, in one of their common ancestors.
- Supports **parallelism with Web Workers**, if available. Disabled by default (why?).
- Very easy to extend without modifying the code, due to Prism's plugin architecture. Multiple hooks are scattered throughout the source.
- Very easy to define new languages. Only thing you need is a good understanding of regular expressions
- All styling is done through CSS, with sensible class names rather than ugly namespaced abbreviated nonsense.
- Wide browser support: IE9+, Firefox, Chrome, Safari, Opera, most Mobile browsers
- Highlights embedded languages (e.g. CSS inside HTML, JavaScript inside HTML)
- Highlights inline code as well, not just code blocks
- Highlights nested languages (CSS in HTML, JavaScript in HTML)
- It doesn't force you to use any Prism-specific markup, not even a Prism-specific class name, only standard markup you should be using anyway. So, you can just try it for a while, remove it if you don't like it and leave no traces behind.
- Highlight specific lines and/or line ranges (requires plugin)
- Show invisible characters like tabs, line breaks etc (requires plugin)
- Autolink URLs and emails, use Markdown links in comments (requires plugin)

Basic usage

You need to include the `prism.css` and `prism.js` authoring files you downloaded in your page. Example:

```html
<!DOCTYPE html>
<html>
<head>
 ...
 <link href="themes/prism.css" rel="stylesheet" />
 ></head>
<body>
 ...
 <script src="prism.js"></script>
 </body>
</html>
```

Prism does its best to encourage good practices. Therefore, it only works with **<code>** elements, since marking up code without a *<code>* element is semantically invalid. According to the HTML5 spec, the recommended way to define a code language is a `language-xxxx` class, which is what Prism uses. Alternatively, Prism also supports a shorter version: `lang-xxxx`.

To make things easier however, Prism assumes that this language definition is inherited. Therefore, if multiple `<code>` elements have the same language, you can add the `language-xxxx` class on one of their common ancestors. This way, you can also define a document-wide default language, by adding a `language-xxxx` class on the `<body>` or `<html>` element.

If you want to opt-out of highlighting for a `<code>` element that is a descendant of an element with a declared code language, you can add the class `language-none` to it (or any non-existing language, really).

The recommended way to mark up a code block (both for semantics and for Prism) is a `<pre>` element with a `<code>` element inside, like so:

```html
<pre><code class="language-css">p { color: red }</code></pre>
```

If you use that pattern, the `<pre>` will automatically get the `language-xxxx` class (if it doesn't already have it) and will be styled as a code block.

If you want to prevent any elements from being automatically highlighted, you can use the attribute **data-manual** on the **<script>** element you used for prism and use the API. Example:

```
<script src="prism.js" data-manual></script>
```

Usage with Webpack, Browserify, & Other Bundlers

If you want to use Prism with a bundler, install Prism with **npm**:

```
$ npm install prismjs
```

You can then **import** into your bundle:

```
import Prism from 'prismjs';
```

To make it easy to configure your Prism instance with only the languages and plugins you need, use the babel plugin, babel-plugin-prismjs. This will allow you to load the minimum number of languages and plugins to satisfy your needs. See that plugin's documentation for configuration details.

Usage with Node

If you want to use Prism on the server or through the command line, Prism can be used with Node.js as well. This might be useful if you're trying to generate static HTML pages with highlighted code for environments that don't support browser-side JS, like AMP pages.

Example:

```
var Prism = require('prismjs');

// The code snippet you want to highlight, as a string
var code = "var data = 1;";

// Returns a highlighted HTML string
var html = Prism.highlight(code, Prism.languages.javascript, 'javascript');
```

Requiring **prismjs** will load the default languages: **markup**, **css**, **clike** and **javascript**. You can load more languages with the **loadLanguages()** utility, which will automatically handle any required dependencies.

Example:

```
var Prism = require('prismjs');
var loadLanguages = require('prismjs/components/');
loadLanguages(['haml']);

// The code snippet you want to highlight, as a string
var code = "= ['hi', 'there', 'reader!'].join \" \"";

// Returns a highlighted HTML string
var html = Prism.highlight(code, Prism.languages.haml, 'haml');
```

Note: Do *not* use **loadLanguages()** with Webpack or another bundler, as this will cause Webpack to include all languages and plugins. Use the babel plugin described above.