10th November 2019

Polyfills

The JavaScript language steadily evolves. New proposals to the language appear regularly, they are analyzed and, if considered worthy, are appended to the list at https://tc39.github.io/ecma262/ and then progress to the specification.

Teams behind JavaScript engines have their own ideas about what to implement first. They may decide to implement proposals that are in draft and postpone things that are already in the spec, because they are less interesting or just harder to do.

So it's quite common for an engine to implement only the part of the standard.

A good page to see the current state of support for language features is https://kangax.github.io/compat-table/es6/ (it's big, we have a lot to study yet).

Babel

When we use modern features of the language, some engines may fail to support such code. Just as said, not all features are implemented everywhere.

Here Babel comes to the rescue.

Babel is a transpiler. It rewrites modern JavaScript code into the previous standard.

Actually, there are two parts in Babel:

1. First, the transpiler program, which rewrites the code. The developer runs it on their own computer. It rewrites the code into the older standard. And then the code is delivered to the website for users. Modern project build systems like webpack provide means to run transpiler automatically on every code change, so that it's very easy to integrate into development process.
2. Second, the polyfill.

   New language features may include new built-in functions and syntax constructs. The transpiler rewrites the code, transforming syntax constructs into older ones. But as for new built-in functions, we need to implement them. JavaScript is a highly dynamic language, scripts may add/modify any functions, so that they behave according to the modern standard.

   A script that updates/adds new functions is called "polyfill". It "fills in" the gap and adds missing implementations.

   Two interesting polyfills are:

   - core js that supports a lot, allows to include only needed features.
   - polyfill.io service that provides a script with polyfills, depending on the features and user's browser.

So, if we're going to use modern language features, a transpiler and a polyfill are necessary.

Examples in the tutorial

Most examples are runnable at-place, like this:

```
1  alert('Press the "Play" button in the upper-right corner to run');
```

Examples that use modern JS will work only if your browser supports it.

Google Chrome is usually the most up-to-date with language features, good to run bleeding-edge demos without any transpilers, but other modern browsers also work fine.

Previous lesson Next lesson

Share

Tutorial map

Comments

- If you have suggestions what to improve - please submit a GitHub issue or a pull request instead of commenting.
- If you can't understand something in the article – please elaborate.
- To insert a few words of code, use the `<code>` tag, for several lines – use `<pre>`, for more than 10 lines – use a sandbox (plnkr, JSBin, codepen…)