@babel/plugin-proposal-class-properties

EDIT

Example

Below is a class with four class properties which will be transformed.

JavaScript Copy

```
class Bork {
  //Property initializer syntax
  instanceProperty = "bork";
  boundFunction = () => {
    return this.instanceProperty;
  };

  //Static class properties
  static staticProperty = "babelIsCool";
  static staticFunction = function() {
    return Bork.staticProperty;
  };
}

let myBork = new Bork;

//Property initializers are not on the prototype.
console.log(myBork.__proto__.boundFunction); // > undefined

//Bound functions are bound to the class instance.
console.log(myBork.boundFunction.call(undefined)); // > "bork"

//Static function exists on the class.
console.log(Bork.staticFunction()); // > "babelIsCool"
```

Installation

Shell Copy

```
npm install --save-dev @babel/plugin-proposal-class-properties
```

Usage

Via `.babelrc` (Recommended)

.babelrc

Without options:

JSON Copy

```
{
  "plugins": ["@babel/plugin-proposal-class-properties"]
}
```

With options:

JSON Copy

```
{
  "plugins": [
    ["@babel/plugin-proposal-class-properties", { "loose": true }]
  ]
}
```

Via CLI

Shell Copy

```
babel --plugins @babel/plugin-proposal-class-properties script.js
```

Via Node API

JavaScript Copy

```
require("@babel/core").transform("code", {
  plugins: ["@babel/plugin-proposal-class-properties"]
});
```

Options

`loose`

`boolean`, defaults to `false`.

When `true`, class properties are compiled to use an assignment expression instead of `Object.defineProperty`.

For an explanation of the consequences of using either, see Definition vs. Assignment (TL;DR in Part 5)

Example

JavaScript Copy

```
class Bork {
  static a = 'foo';
  static b;

  x = 'bar';
  y;
}
```

Without `{ "loose": true }`, the above code will compile to the following, using `Object.defineProperty`:

JavaScript Copy

```
var Bork = function Bork() {
  babelHelpers.classCallCheck(this, Bork);
  Object.defineProperty(this, "x", {
    configurable: true,
    enumerable: true,
    writable: true,
    value: 'bar'
  });
  Object.defineProperty(this, "y", {
    configurable: true,
    enumerable: true,
    writable: true,
    value: void 0
  });
};

Object.defineProperty(Bork, "a", {
  configurable: true,
  enumerable: true,
  writable: true,
  value: 'foo'
});
Object.defineProperty(Bork, "b", {
  configurable: true,
  enumerable: true,
  writable: true,
  value: void 0
});
```

However, with `{ "loose": true }`, it will compile using assignment expressions:

JavaScript Copy

```
var Bork = function Bork() {
  babelHelpers.classCallCheck(this, Bork);
  this.x = 'bar';
  this.y = void 0;
};

Bork.a = 'foo';
Bork.b = void 0;
```

> You can read more about configuring plugin options here

References

- Proposal: Public and private instance fields
- Proposal: Static class features