C input source	Linear assembly bytecode (intermediate representation)	Wasm binary encoding (hexadecimal bytes)

`int factorial(int n) {`
`  if (n == 0)`
`  return 1;`
`  else`
`  return n * factorial(n-1);`
`}`
	`get_local 0`
	`i64.eqz`
	`if (result i64)`
	`  i64.const 1`
	`else`
	`  get_local 0`
	`  get_local 0`
	`  i64.const 1`
	`  i64.sub`
	`  call 0`
	`  i64.mul`
	`end`
		`20 00`
		`50`
		`04 7E`
		`42 01`
		`05`
		`20 00`
		`20 00`
		`42 01`
		`7D`
		`10 00`
		`7E`
		`0B`

The WebAssembly text format can also be written in a folded format using s-expressions. This format is purely syntactic sugar and has no behavioral differences with the linear format.[58] An example is shown below:

```
(module
  (import "math" "exp" (func $exp (param f64) (result f64)))
  (func (export "doubleExp") (param $0 f64) (result f64)
    (f64.mul
      (call $exp
        (get_local $0)
      )
      (f64.const 2)
    )
  )
)
```

babel-preset-php

PHP7 to ES7 syntax translator

This project transpiles PHP source code to readable JavaScript source code.

The conversion is implemented as an AST to AST translation (with source maps!). Produced code will have valid JavaScript syntax, but may not work quite the same way due to many conceptual differences between the languages.

PHP input →

```
define('FOO', max(floatval($c), strlen("foo")));
$bar['x'][][$b] = json_encode(__FILE__);
class Foo extends Bar\Baz {
  var $z = "hello" . "world";
  function __construct($some = array(7)) {
    parent::__construct(func_get_args());
    self::${$k} = "{$this->z[10]}";
  }
}
```

→ generated JS

```
const FOO = Math.max(+c, "foo".length);
bar.x.push({[b]: JSON.stringify(__filename)});
class Foo extends Bar.Baz {
  constructor(some = [7]) {
    super(arguments);
    this.z = "hello" + "world";
    this.constructor[k] = `${this.z[10]}`;
  }
}
```