

```

    name: "y"
  }
],
"body": {
  "type": "Identifier",
  "name": "y"
}
}
]

```

We choose to represent Yocto-JavaScript programs with the data structures above because they follow a specification called ESTree [1], and by adhering to this specification we may reuse tools from the JavaScript ecosystem (see § 1.3.17 and § 1.3.18).

In general, the data structures used to represent Yocto-JavaScript programs are of the following types (written as TypeScript types adapted from the ESTree types [3] to include only the features supported by Yocto-JavaScript):

```

type Expression = ArrowFunctionExpression | CallExpression | Identifier;

```

```

interface ArrowFunctionExpression {
  type: "ArrowFunctionExpression";
  params: [Identifier];
  body: Expression;
}

```

```

interface CallExpression {
  type: "CallExpression";
  callee: Expression;
  arguments: [Expression];
}

```

```

interface Identifier {
  type: "Identifier";
  name: string;
}

```

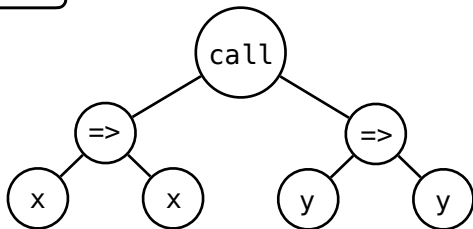
(The definitions above correspond to elements of the Yocto-JavaScript grammar (see § 1.1.4); for example, `Expression` corresponds to  $e$ .)

In later steps various aspects of the interpreter will change, including parts of

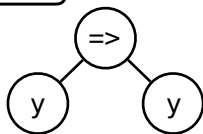
evaluate()

"(x => x)(y => y)"

parse()

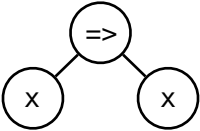


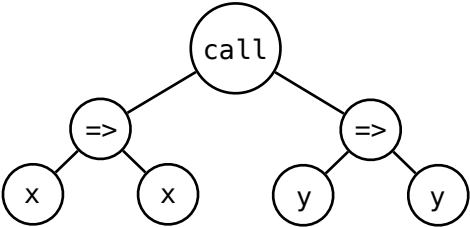
run()



prettify()

"y => y"

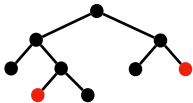




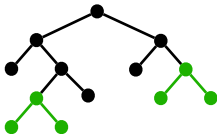
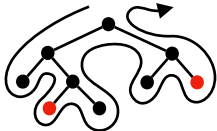
parameter

body

argument



substitute()

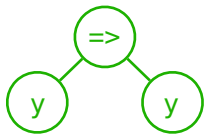


Traversal

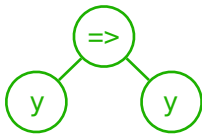
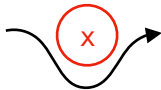
parameter

body

argument



substitute()

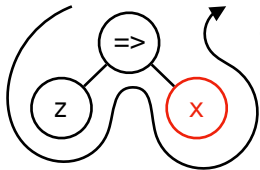
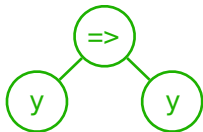
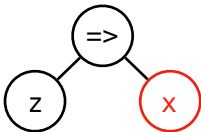


**Traversal**

parameter

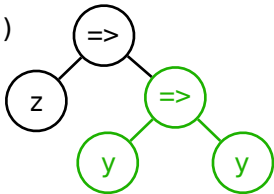
body

argument



**Traversal**

`substitute()`

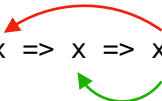


# Option 1: No shadowing

Refers to

Result

$(x \Rightarrow x \Rightarrow x) (y \Rightarrow y)$



$x \Rightarrow y \Rightarrow y$

$x \Rightarrow x$

# Option 2: Shadowing