CIVET

The Modern Way to Write TypeScript

Expressive Syntax and Faster Coding with Civet

Cheatsheet   Civet playground

Civet is a programming language that compiles to TypeScript or JavaScript, so you can use existing tooling but enable concise and powerful syntax. In addition to 99% JS/TS compatibility, there are many features, with some highlights below and more comprehensive examples on the cheatsheet. See also Civet's design philosophy.

Highlights: Beyond TC39

Civet code on top, compiled TypeScript output on bottom.

Pattern Matching

TC39 Proposal: Pattern Matching

```
switch x
  0
    console.log("zero")
  /^\s+$/
    console.log("whitespace")
  [{type: "text", content}, ...rest]
    console.log("leading text", content)
```

Edit inline or edit in the Playground!

```
if (x === 0) {
  console.log("zero");
} else if (
  typeof x === "string" &&
  /^\s+$/.test(x)
) {
  console.log("whitespace");
} else if (
  Array.isArray(x) &&
  x.length >= 1 &&
  typeof x[0] === "object" &&
  x[0] != null &&
  "type" in x[0] &&
  x[0].type === "text" &&
  "content" in x[0]
) {
  const [{ type, content }, ...rest] = x;
  console.log("leading text", content);
}
```

▨ Ligatures

Pipelines

TC39 Proposal: Pipe Operator

```
data
  |> Object.keys
  |> console.log
```

Edit inline or edit in the Playground!

```
console.log(Object.keys(data));
```

▨ Ligatures

Pipe expression with shorthand functions:

```
a |> & + 1 |> bar
```

Edit inline or edit in the Playground!

```
bar(a + 1);
```

▨ Ligatures

Single-Argument Function Shorthand

```
x.map .name
x.map &.profile?.name[0...3]
x.map &.callback a, b
x.map +&
```

Edit inline or edit in the Playground!

```
x.map(($) => $.name);
x.map(($1) => $1.profile?.name.slice(0, 3));
x.map(($2) => $2.callback(a, b));
x.map(($3) => +$3);
```

▨ Ligatures

Custom Infix Operators

```
operator {min, max} := Math
value min ceiling max floor
```

Edit inline or edit in the Playground!

```
const { min, max } = Math;
max(min(value, ceiling), floor);
```

▨ Ligatures

Everything is an Expression

```
items = for item of items
  if item.length
    item.toUpperCase()
  else
    "<empty>"
```

Edit inline or edit in the Playground!

```
items = (() => {
  const results = [];
  for (const item of items) {
    if (item.length) {
      results.push(item.toUpperCase());
    } else {
      results.push("<empty>");
    }
  }
  return results;
})();
```

▨ Ligatures

```
return
  if x == null
    throw "x is null"
  else
    log `received x of ${x}`
    x.value()
```

Edit inline or edit in the Playground!

```
return x == null
  ? (() => {
      throw "x is null";
    })()
  : (log(`received x of ${x}`), x.value());
```

▨ Ligatures

TC39 proposal: `do` expressions

```
x = do
  const tmp = f()
  tmp * tmp + 1
```

Edit inline or edit in the Playground!

```
x = (() => {
  {
    const tmp = f();
    return tmp * tmp + 1;
  }
})();
```

⬛ Ligatures

Dedented Strings and Templates

TC39 Proposal: String Dedent

```
text = """
  This text is a string that doesn't include
  the leading whitespace.
"""
```

Edit inline or edit in the Playground!

```
text = `This text is a string that doesn't include
the leading whitespace.`;
```

⬛ Ligatures

```
text = ```
  Also works for
  ${templates}!
```
```

Edit inline or edit in the Playground!

```
text = `Also works for
${templates}!`;
```

⬛ Ligatures
```

Chained Comparisons

```
a < b <= c
a is b is not c
a instanceof b not instanceof c
```

Edit inline or edit in the Playground!

```
a < b && b <= c;
a === b && b !== c;
a instanceof b && !(b instanceof c);
```

▨ Ligatures

Default to `const` for Iteration Items

```
for (item of [1, 2, 3, 4, 5]) {
  console.log(item * item);
}
```

Edit inline or edit in the Playground!

```
for (const item of [1, 2, 3, 4, 5]) {
  console.log(item * item);
}
```

▨ Ligatures

Spread in Any Position

Spreads in first or middle position:

```
[...head, last] = [1, 2, 3, 4, 5]
```

Edit inline or edit in the Playground!

```
const splice: <T>(
  this: T[],
  start: number,
  deleteCount?: number
) => T[] = [].splice as any;
([...head] = [1, 2, 3, 4, 5]),
  ([last] = splice.call(head, -1));
```

Ligatures

```
{a, ...rest, b} = {a: 7, b: 8, x: 0, y: 1}
```

Edit inline or edit in the Playground!

```
({ a, b, ...rest } = { a: 7, b: 8, x: 0, y: 1 });
```

Ligatures

```
function justDoIt(a, ...args, cb) {
  cb.apply(a, args)
}
```

Edit inline or edit in the Playground!

```
const splice: <T>(
  this: T[],
  start: number,
  deleteCount?: number
) => T[] = [].splice as any;
function justDoIt(a, ...args) {
  let [cb] = splice.call(args, -1);
  return cb.apply(a, args);
}
```

Ligatures

Import Syntax Matches Destructuring

```
import {X: LocalX, Y: LocalY} from "./util"
```

Edit inline or edit in the Playground!

```
import { X as LocalX, Y as LocalY } from "./util";
```

Ligatures

Export Convenience

```
export a, b, c from "./cool.js"
export x = 3
```

Edit inline or edit in the Playground!

```
export { a, b, c } from "./cool.js";
export var x = 3;
```

☐ Ligatures

JSX

```
function Listing(props)
  <h1 #heading>Hello Civet!
  <ul .items>
    <For each=props.items>
      (item) =>
        <li .item {props.style}><Item {item}>
```

Edit inline or edit in the Playground!

```
function Listing(props) {
  return (
    <>
      <h1 id="heading">Hello Civet!</h1>
      <ul class="items">
        <For each={props.items}>
          {(item) => {
            return (
              <li
                class="item"
                style={props.style}
              >
                <Item item={item} />
              </li>
            );
          }}
        </For>
      </ul>
    </>
  );
}
```

Ligatures

Sponsors

Thank you to all of our sponsors for your invaluable support and contribution to the Civet language!

builder.io

Support the future development of Civet!

Contributors

Thank you for your work and dedication to the Civet project!

STRd6 (853) edemaine (311) Argeento (21) lorefnon (12)

lamlpanda (4) Milo123459 (2) aleksrutins (1) lrian3x3 (1)

samualtnorman (1) tomByrer (1) zolomatok (1)

Edit this page on GitHub

Last updated: 2/26/2023, 2:48:31AM