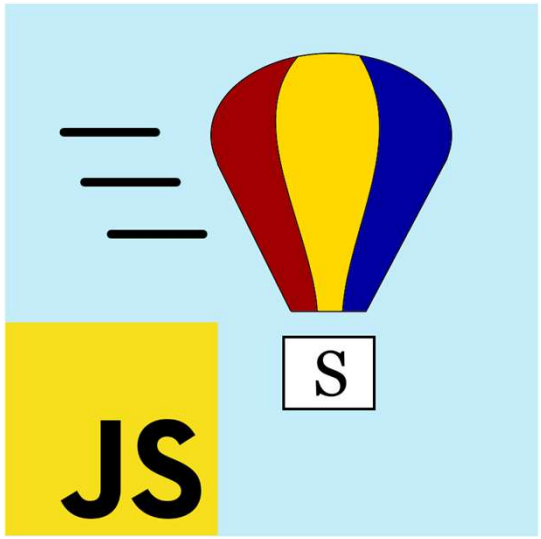
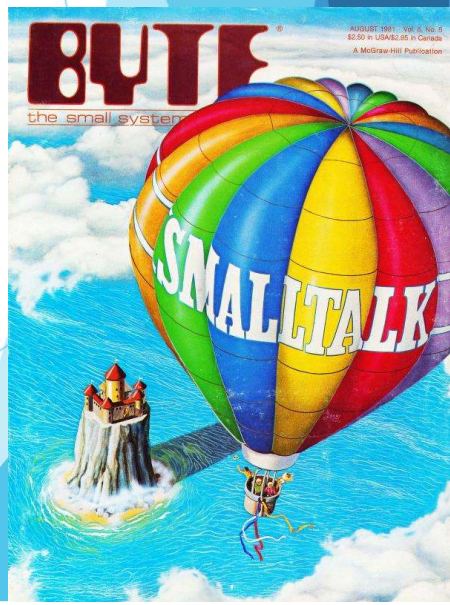
SmallJS

Back to elegance



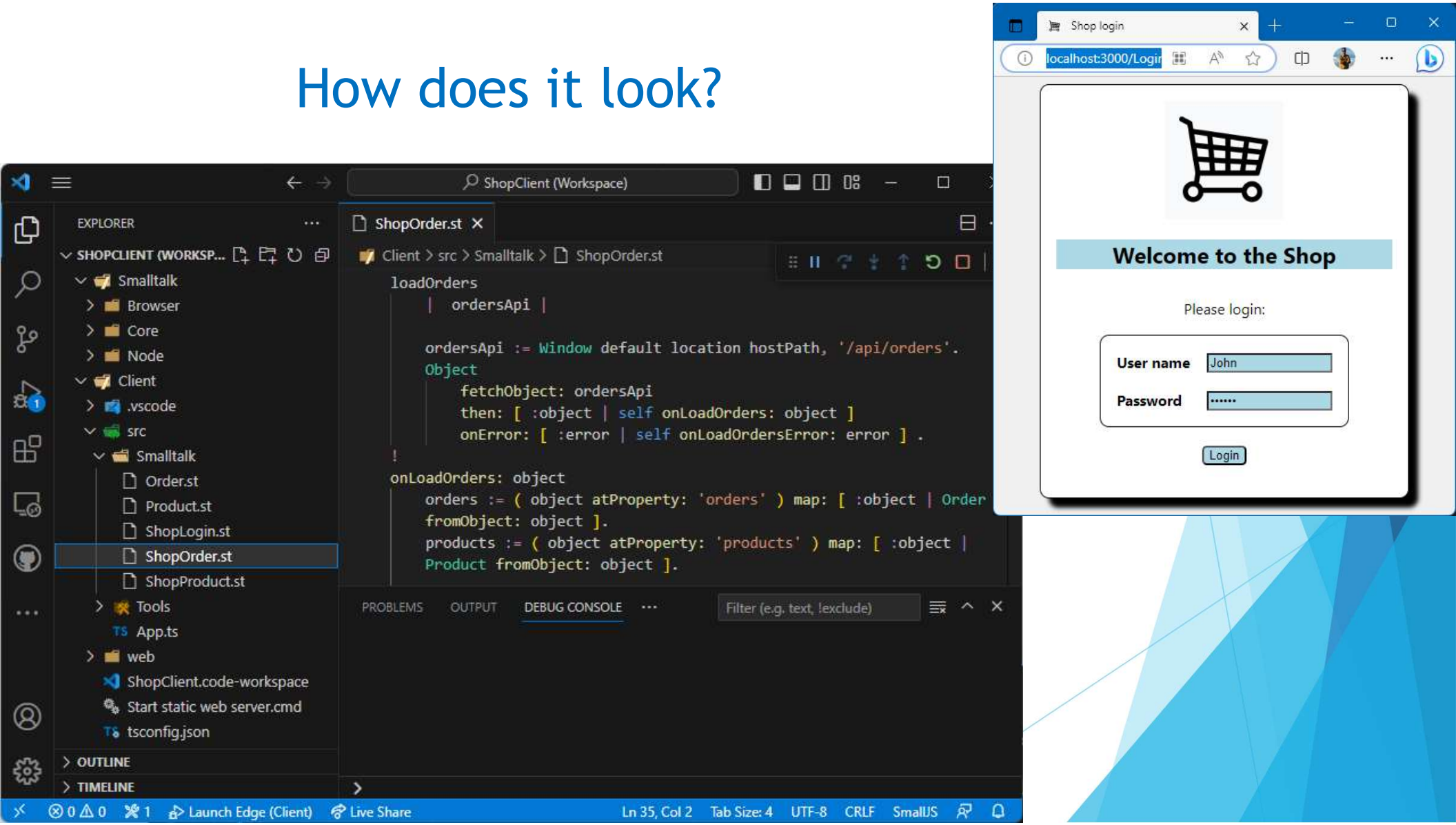


Website: small-js.org

What is SmallJS?

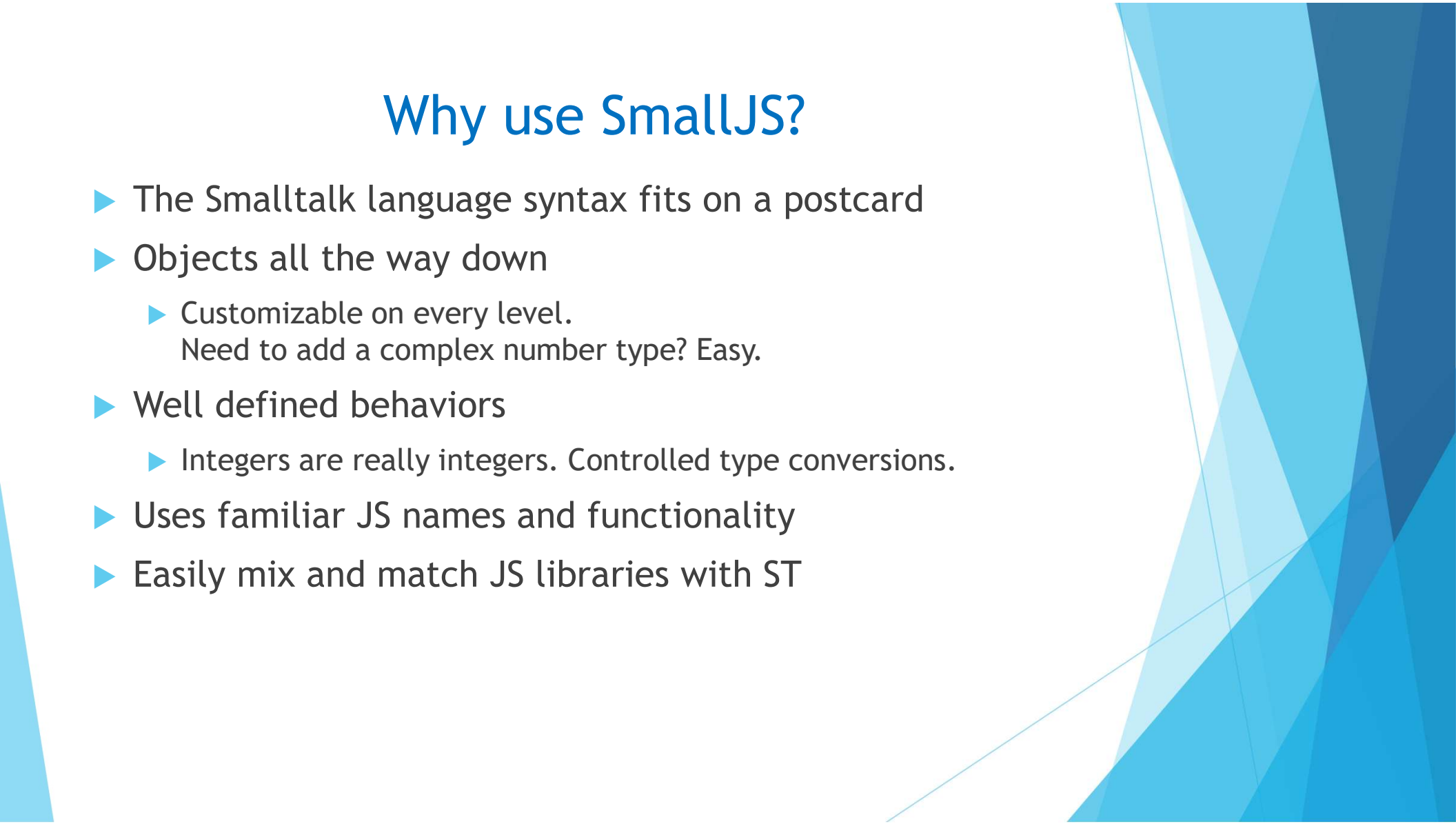
- Transpiler from Smalltalk to efficient, lightweight JavaScript.
- JS code runs in all browsers ***and*** Node.js.
- Smalltalk-80 language syntax support
  - Class and method names like familiar JS.
- Source file based (not image based).
- Development in Visual Studio Code IDE
  - With syntax coloring and step debugging!
- JS libraries already encapsulated in ST:
  - Browser: Document, Window, HTML elements, events, CSS, streams.
  - Node.JS: HTTP server, Express, 3 databases.
- Examples, including a webshop client + server app template.

How does it look?

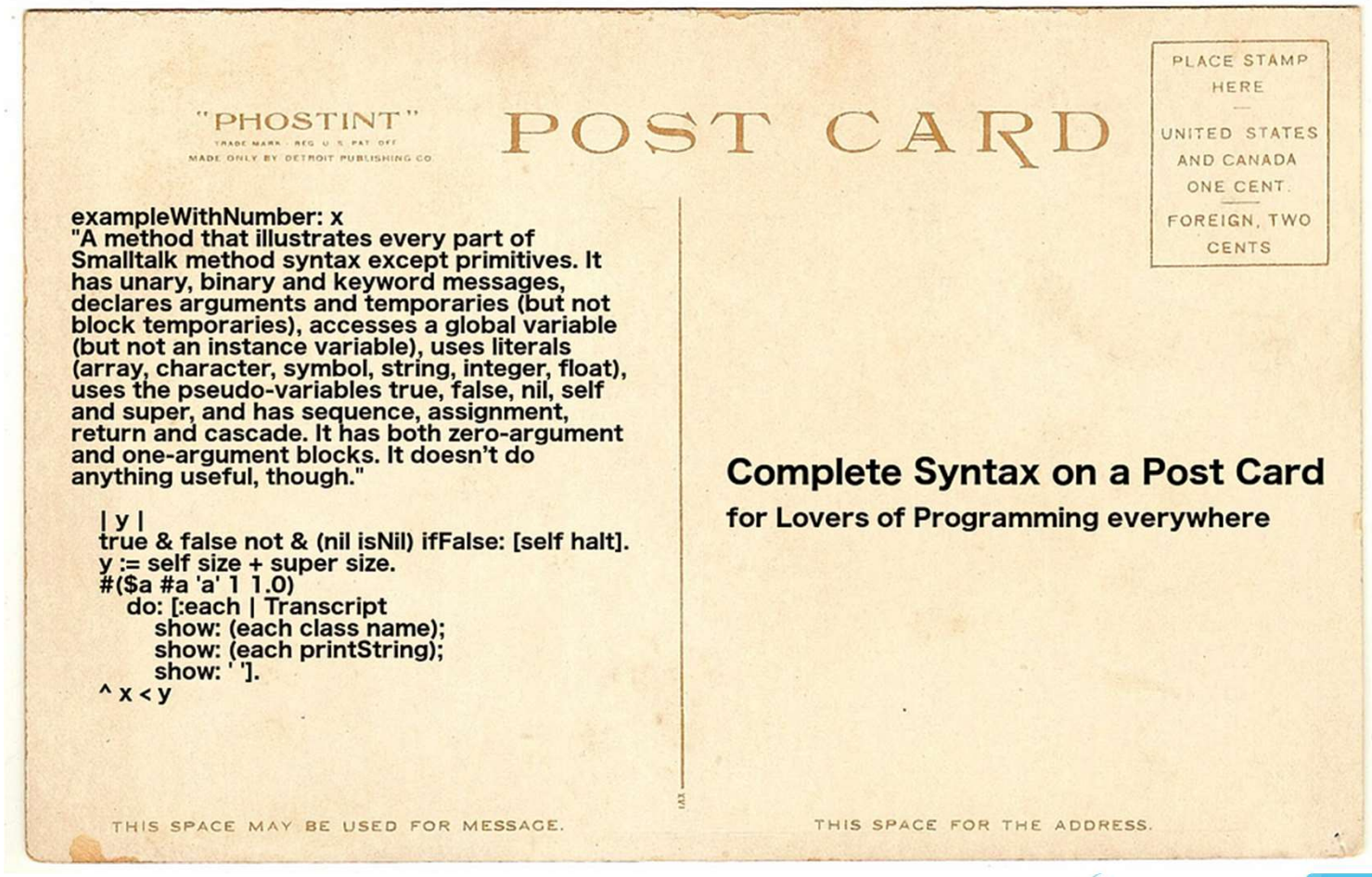


Why use SmallJS?

- The Smalltalk language syntax fits on a postcard
- Objects all the way down
  - Customizable on every level.
    Need to add a complex number type? Easy.
- Well defined behaviors
  - Integers are really integers. Controlled type conversions.
- Uses familiar JS names and functionality
- Easily mix and match JS libraries with ST



Smalltalk syntax on a postcard



Example ST vs JS

SmallJS

Code:

```
squared
  ^ self * self.
```

Call with > result:

```
10 squared          > 100
1.5 squared         > 2.25
( 1 / 3 ) squared   > ( 1 / 9 )
99999999 squared    > right answer, BigInt
‘s’ squared         > error, stops
```

> Number is a base class for Integer, Float, Fraction and BigInt, but not String.

JavaScript

Code:

```
NumUtil.squared( n )
  { return n * n; }
```

Call with > result:

```
NumUtil.squared( 10 )          > 100
NumUtil.squared( 1.5 )         > 2.25
NumUtil.squared( 1 / 3 )       > 0.1111111111111
NumUtil.squared( 99999999 )    > wrong answer, float
NumUtil.squared( ‘s’ )         > NaN, continues
```

> No (safe) integers, no fractions, BigInt not integrated, error prone type coersion.

What about my new JS/TS features?

ST implementation of JS/TS language features.

Feature	ST solution
functional programming	Don't use state vars. Use array iterators.
interface	Abstract base class
Record, tuple	Class with only getters
decorator	Pass anonymous function (block)
static	Class method
private	No getter method
variable argument list	Array
optional argument	Extra 1-line method
import / export	Automatic
async / await	Implemented, unfortunately...
type checking	Build IDE type inferencing (not yet)

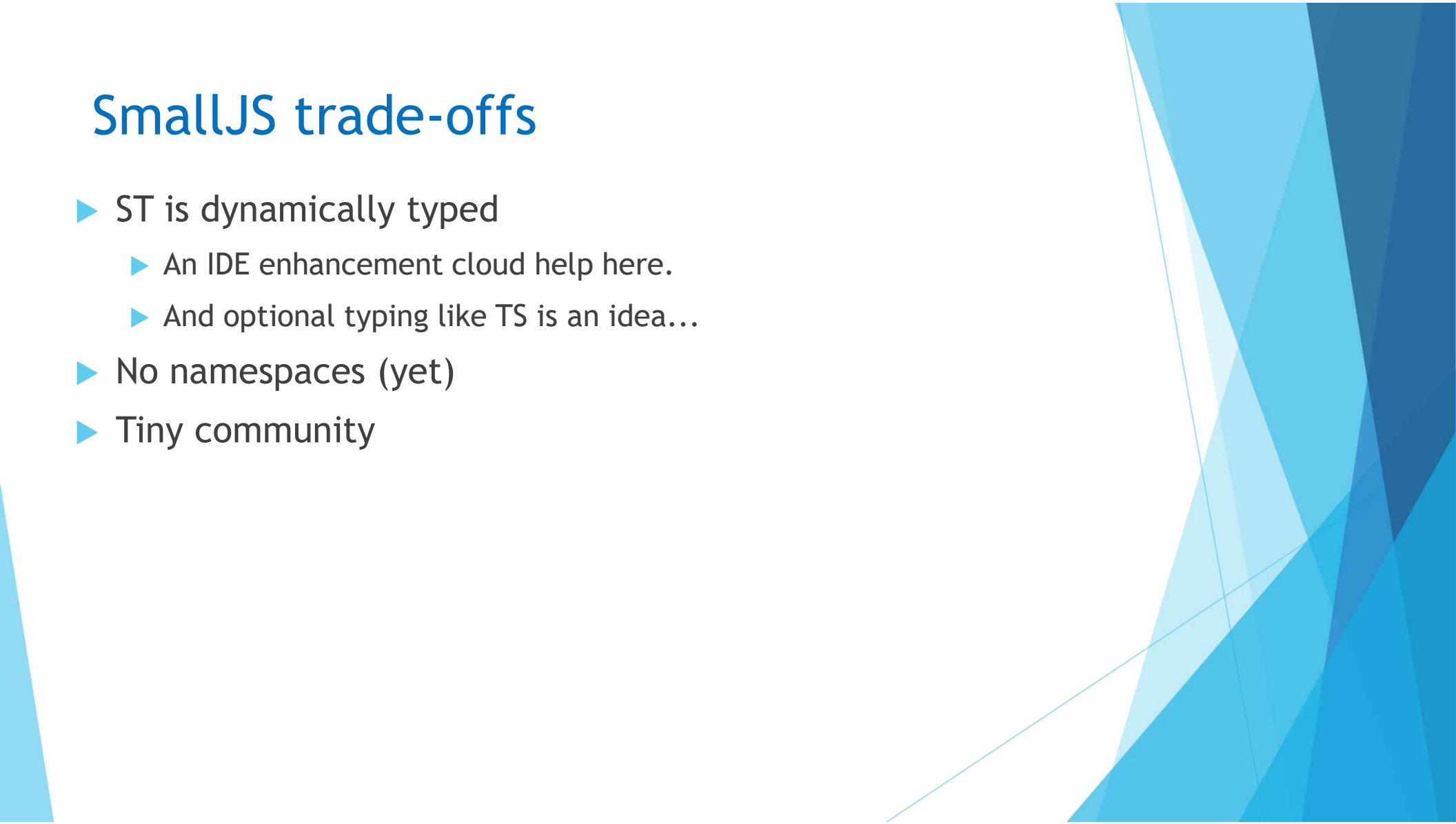
The philosophy is that retaining simplicity is worth some extra lines of encapsulated code.

SmallJS vs traditional Smalltalks
(E.g.: Pharo, Dolphin, Cincom, Squeak)

- File-based (not image-based)
  - Easy source control in clear hierarchy. IDE safely separate from code. Modular class loading iso unsafe image stripping. Can use rich and familiar IDE (VSCode).
- Run anywhere
  - One language for front-end and back-end apps in all browsers and Node.js.
    So also runs on mobile devices.
- Integrates smoothly with the rich JS ecosystem
  - Typically 1 line of interfacing code per encapsulated JS method.
  - ST can call JS and vice versa.
  - Newly available JS features and libraries can be integrated quickly.

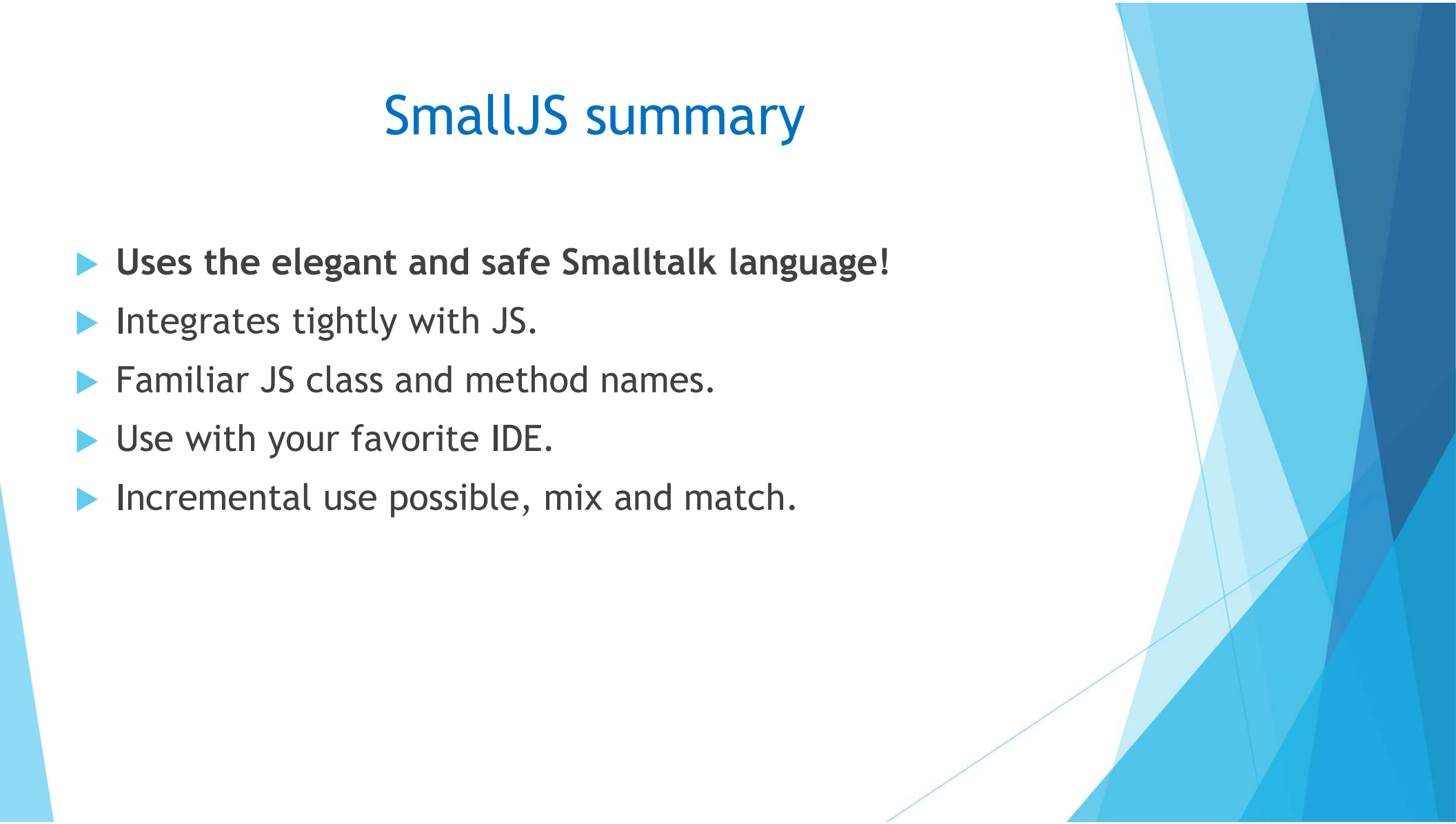
SmallJS trade-offs

- ST is dynamically typed
  - An IDE enhancement cloud help here.
  - And optional typing like TS is an idea...
- No namespaces (yet)
- Tiny community

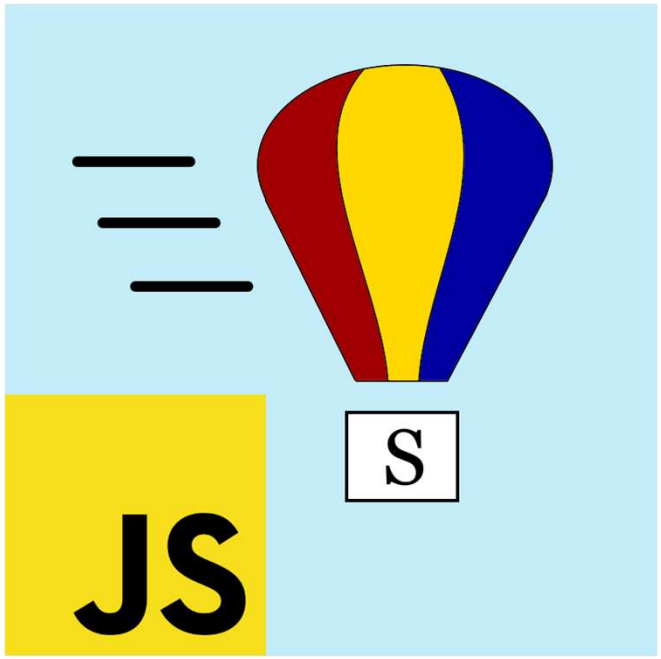


SmallJS summary

- **Uses the elegant and safe Smalltalk language!**
- Integrates tightly with JS.
- Familiar JS class and method names.
- Use with your favorite IDE.
- Incremental use possible, mix and match.



Any questions?



Website: small-js.org