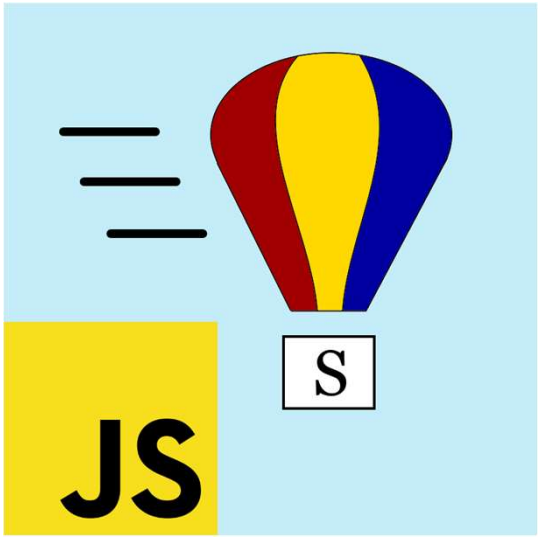
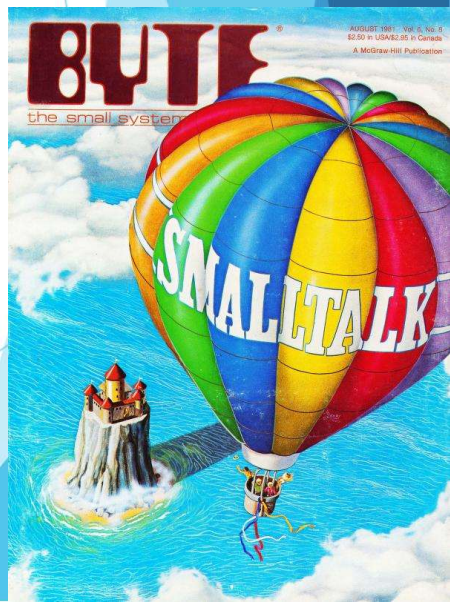
SmallJS

Back to elegance

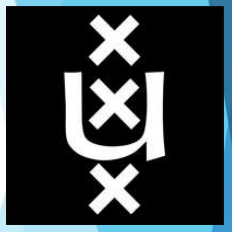
by:
Richard Ronteltap



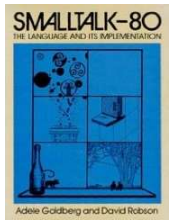


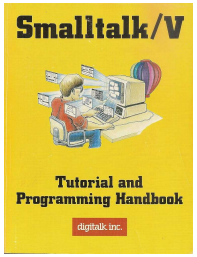
Website: small-js.org

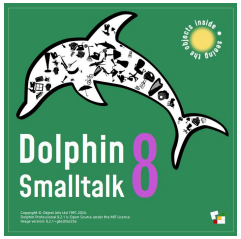
About the author



- My name is **Richard Ronteltap**
  (Yes, that *is* a strange last name, even here :)
- Live in Amsterdam, The Netherlands
- Studied Computer Science at the University of Amsterdam
  where I met my wife (she insisted I say this :),
  where I work as an IT department head.
- My CS thesis in '93 was on OO databases,
  with a query optimizer written in Smalltalk-80.
- I also discovered: Smalltalk/V, Dolphin, Pharo, ...





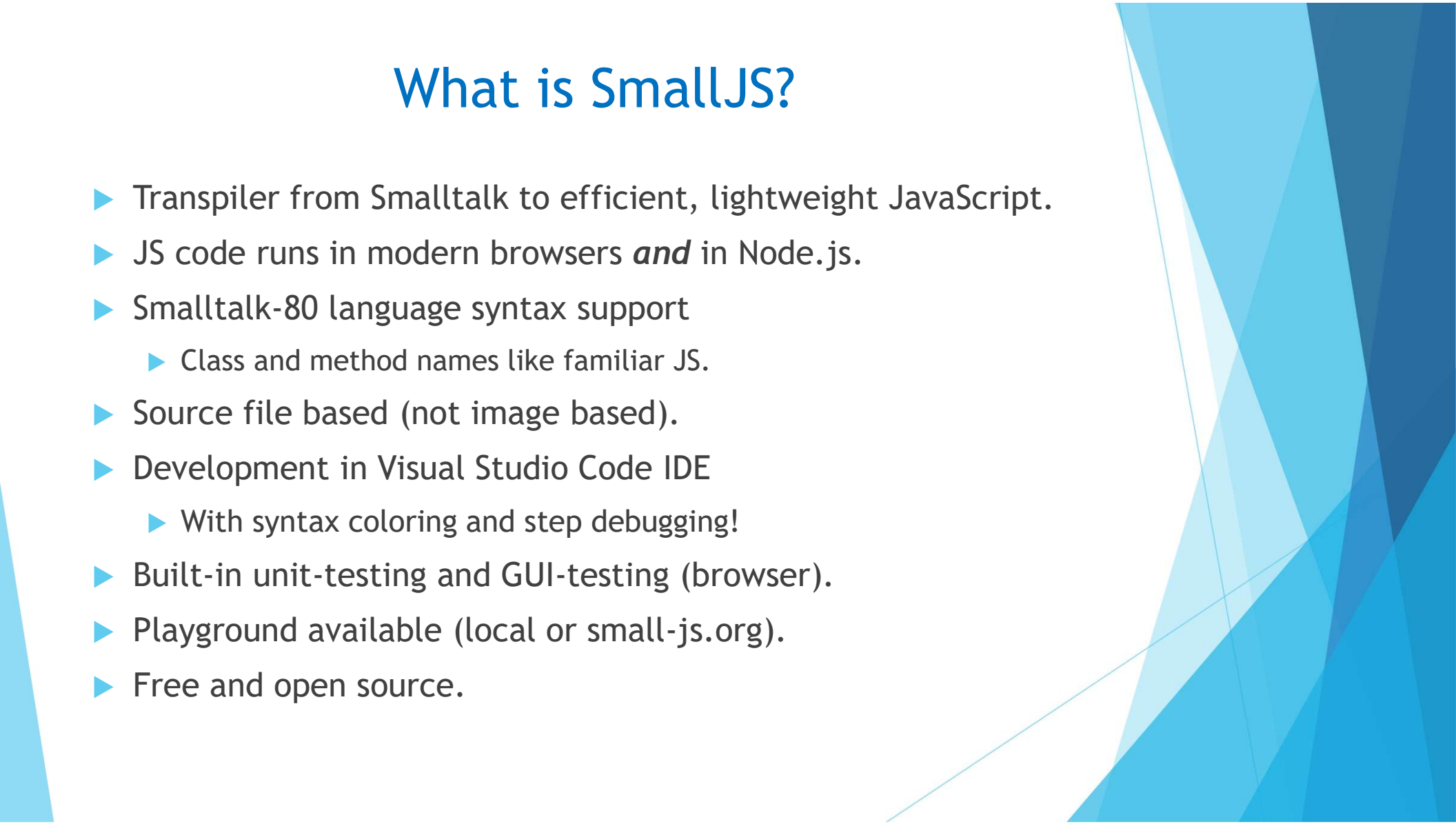


Why make SmallJS?

- Develop modern apps in ST, specifically web apps,
  using the same language for the front-end and the back-end.
- Promote the elegant Smalltalk language
  specifically to JS, PHP and Ruby devs,
  who don't know what they're missing... :)
- Looked at: Amber Smalltalk, PharoJS, Seaside
  But though it could be more JS native and integrated.
- And a ***separate*** IDE and a modular image don't seem so bad...

What is SmallJS?

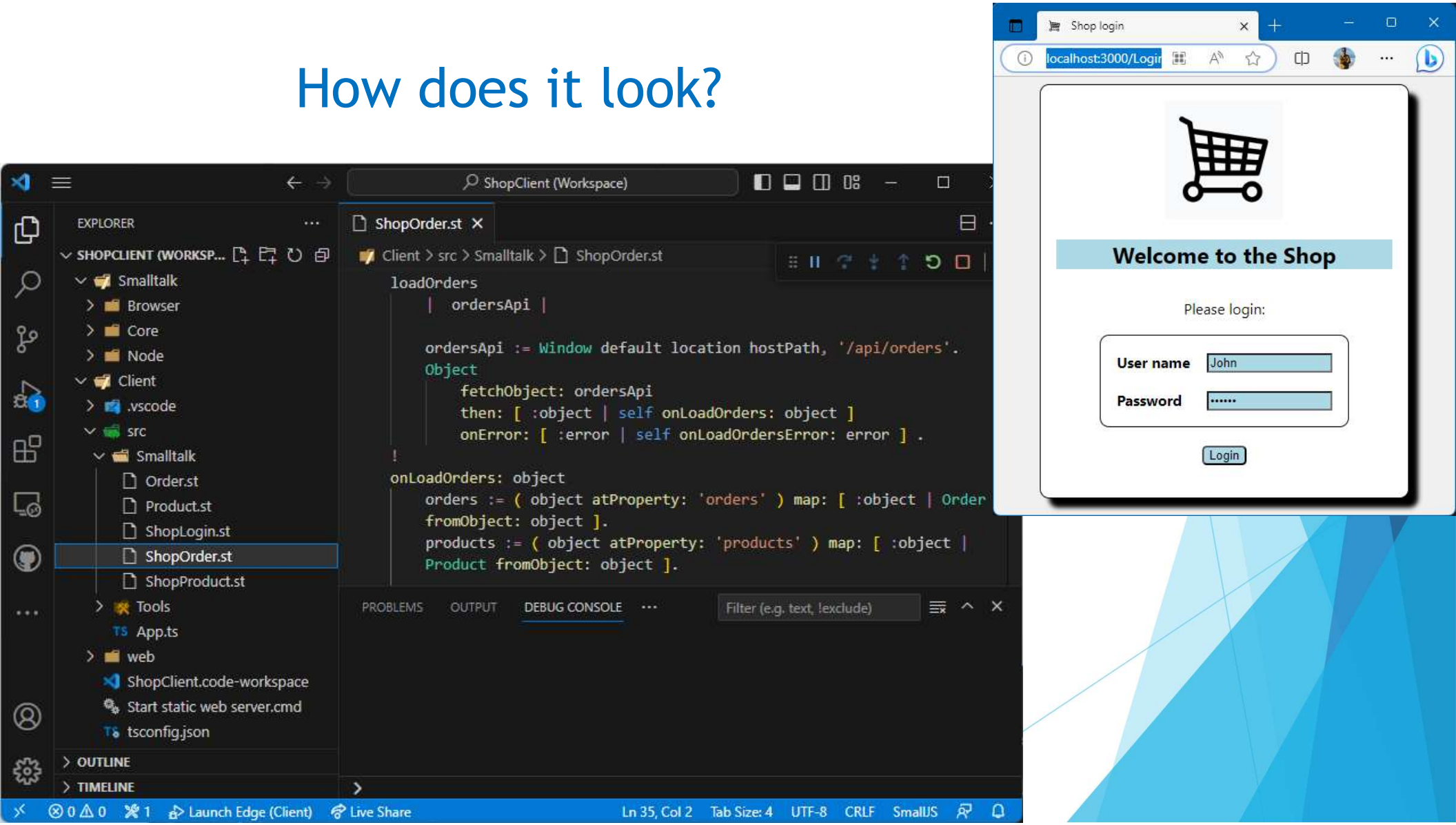
- Transpiler from Smalltalk to efficient, lightweight JavaScript.
- JS code runs in modern browsers ***and*** in Node.js.
- Smalltalk-80 language syntax support
  - Class and method names like familiar JS.
- Source file based (not image based).
- Development in Visual Studio Code IDE
  - With syntax coloring and step debugging!
- Built-in unit-testing and GUI-testing (browser).
- Playground available (local or small-js.org).
- Free and open source.



SmallJS libraries

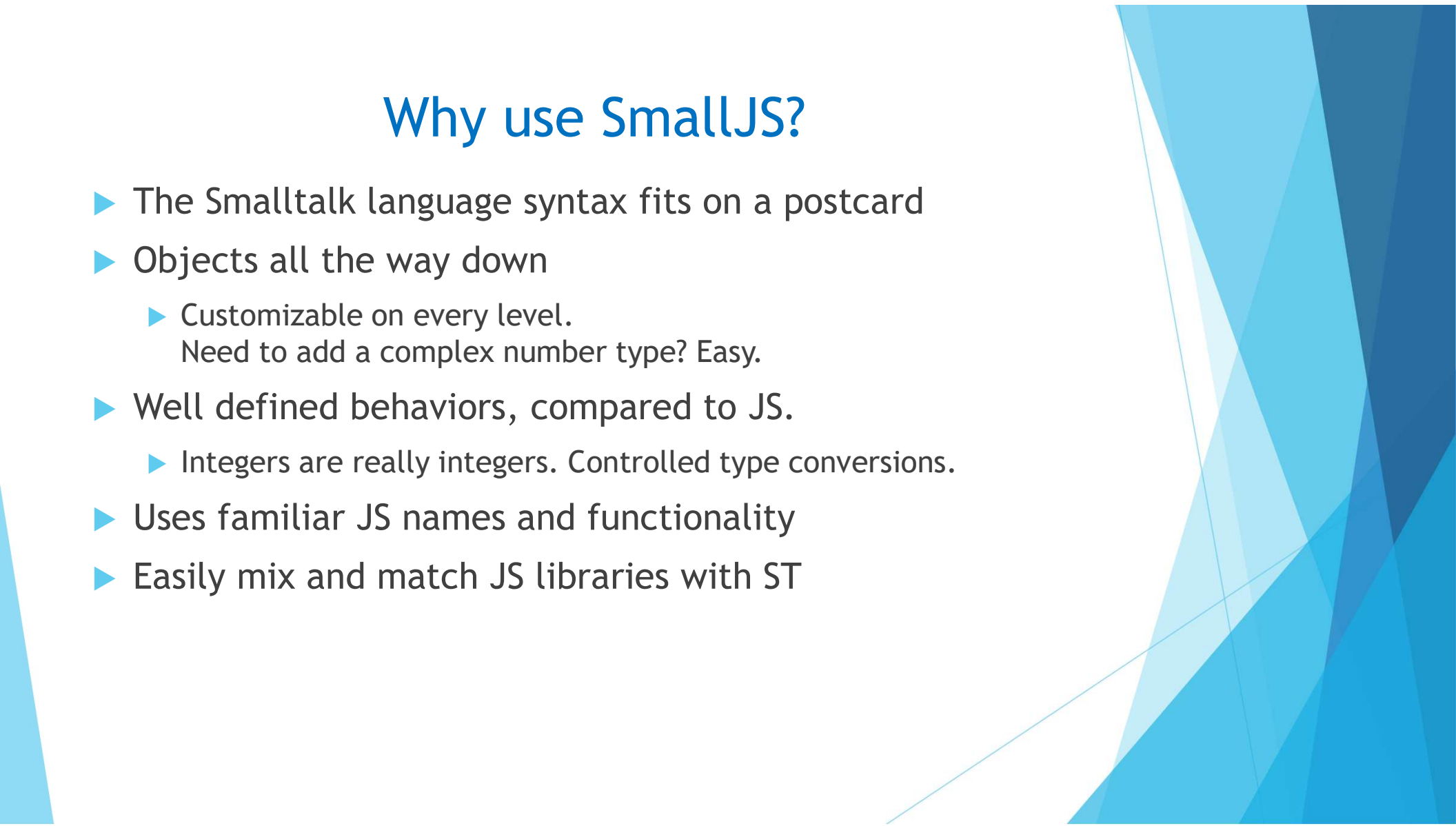
- Core (common)
  - Basic ST classes, full ST magnitude hierarchy.
  - Events, streams, fetch.
- Browser
  - Document, Window, HTML elements, Canvas 2D, CSS, web workers.
  - Lightweight HTML components for SPAs.
- Node.js
  - HTTP server, Express, worker threads, File.
  - Databases: SQLite, Postgres, MariaDB and MySQL with an ORM.
  - AI: OpenAI, Deepseek, 2 more soon..
- Desktop app support: Electron, NodeGui
- Example projects to kick-start using the above.

How does it look?

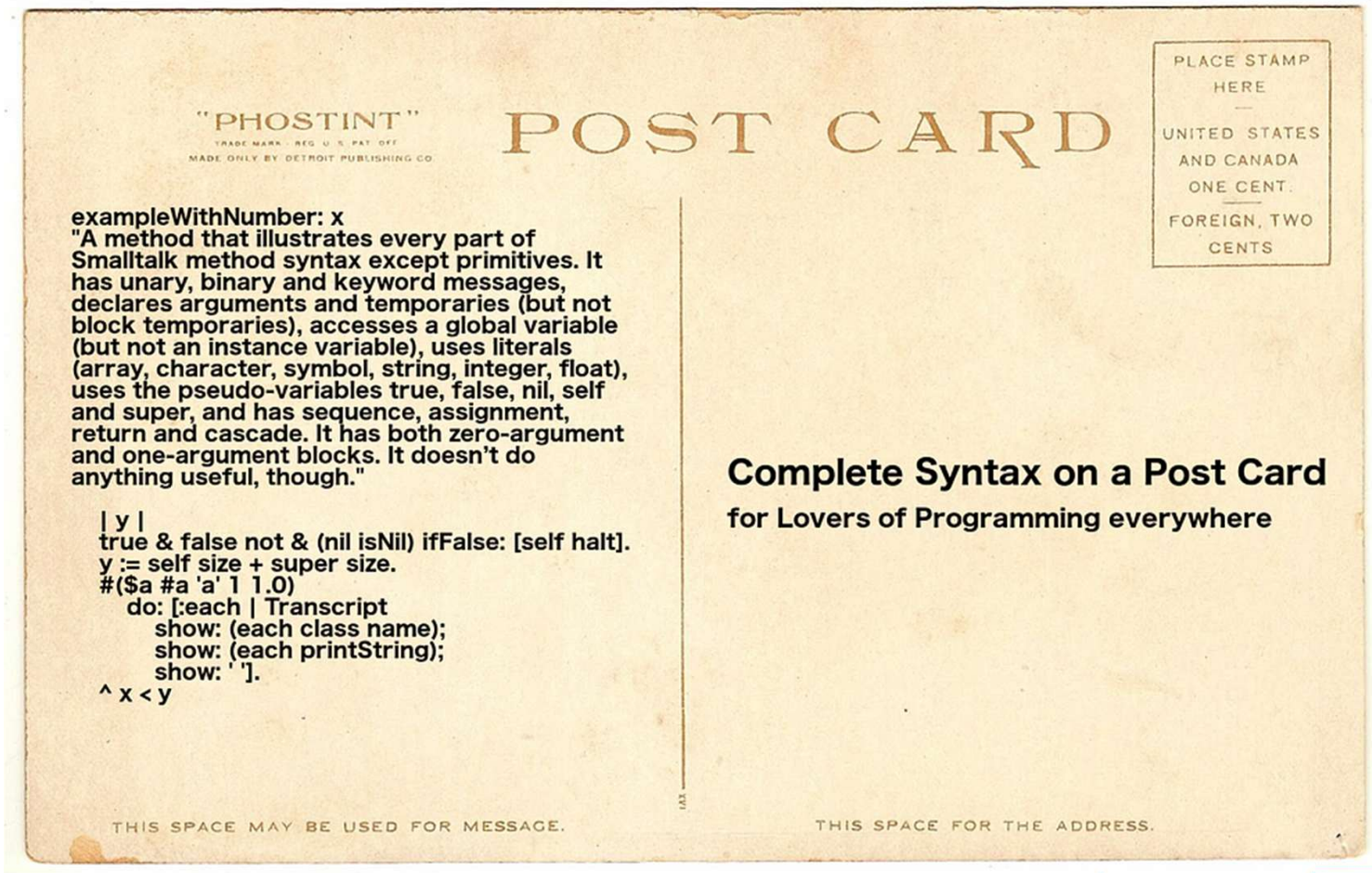


Why use SmallJS?

- 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, compared to JS.
  - Integers are really integers. Controlled type conversions.
- Uses familiar JS names and functionality
- Easily mix and match JS libraries with ST



Smalltalk syntax fits 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)

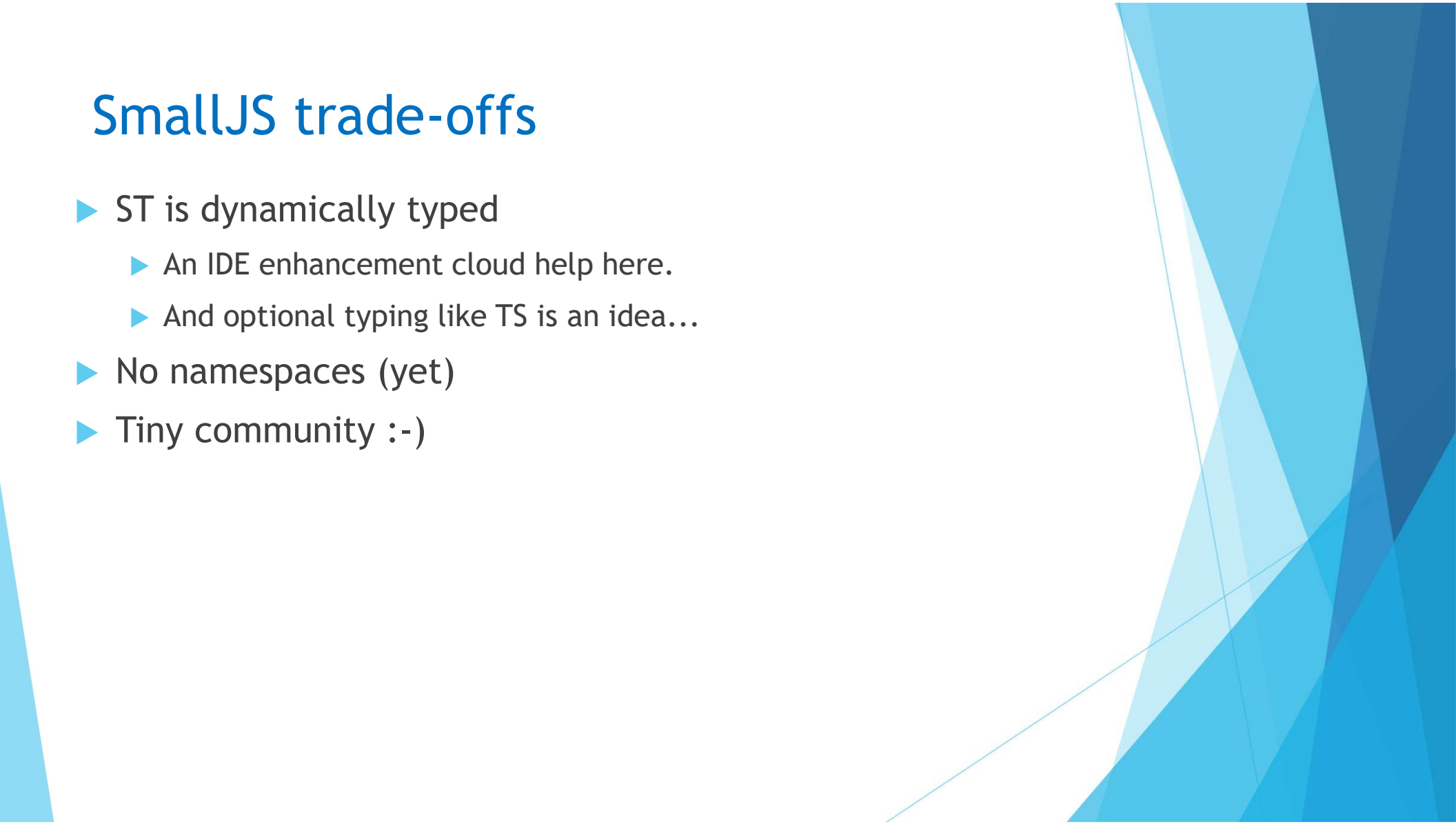
The philosophy is that retaining simplicity is worth some extra lines of encapsulated code.

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.

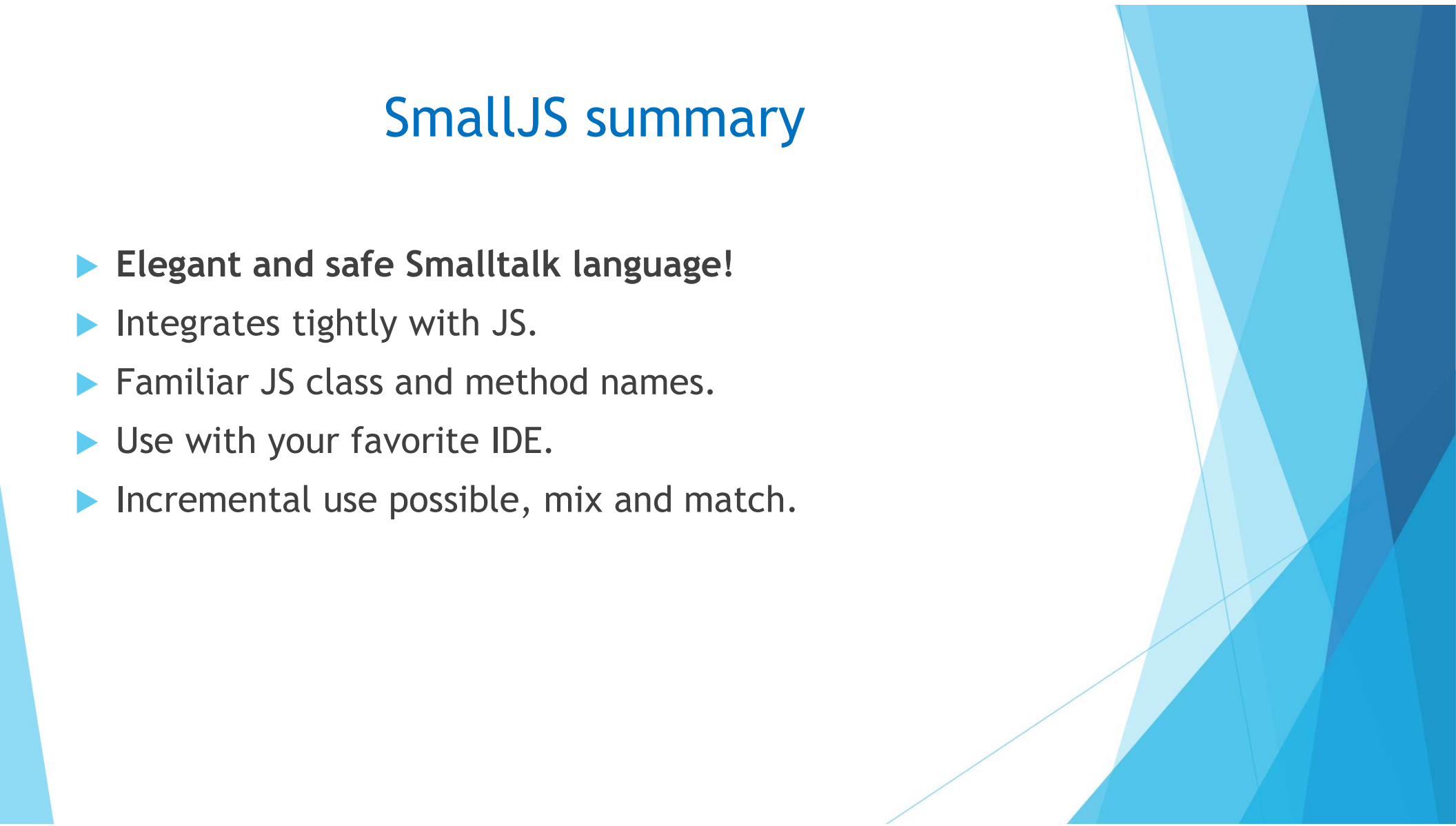
SmallJS trade-offs

- 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

- **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.



Live demo's (basic)

with code inspection

Basic example apps:

App	Notes
Counter	Standard minimal web app.
Todo	Todo list web app with CRUD operations, sorting and multi-language support.
Balls	Colorful bouncing balls, using the HTML Canvas 2D API, which is fully supported.
Shop	Basic web shop front-ends, traditional and as SPA and back-end, with database support on SQL ORM mapper.

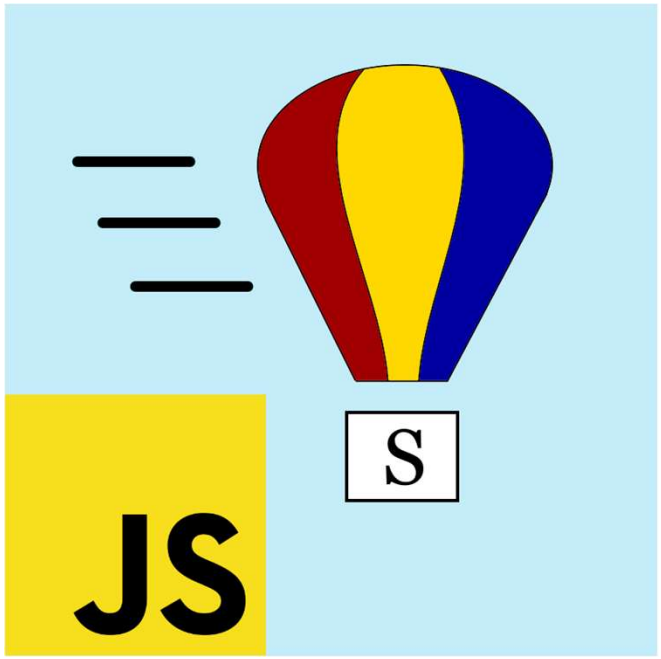
Live demo's (advanced)

with code inspection

Advanced example apps:

App	Notes
Benchmark	Number crunching in SmallJS for: primes, Fibonacci, Pi
WebWorkers	True multithreading using the Web Workers API. (Node worker threads are also supported)
Electron	Platform independent desktop app based on Electron (when you don't care about memory usage and complexity :)
NodeGui	Platform independent desktops app based on NodeGui/QT (when you ***do*** care about memory usage and simplicity)
Pharo	SmallJS front-end with Pharo back-end using Zinc and API call. (So quite basic on both sides)

Any questions?



Website: small-js.org