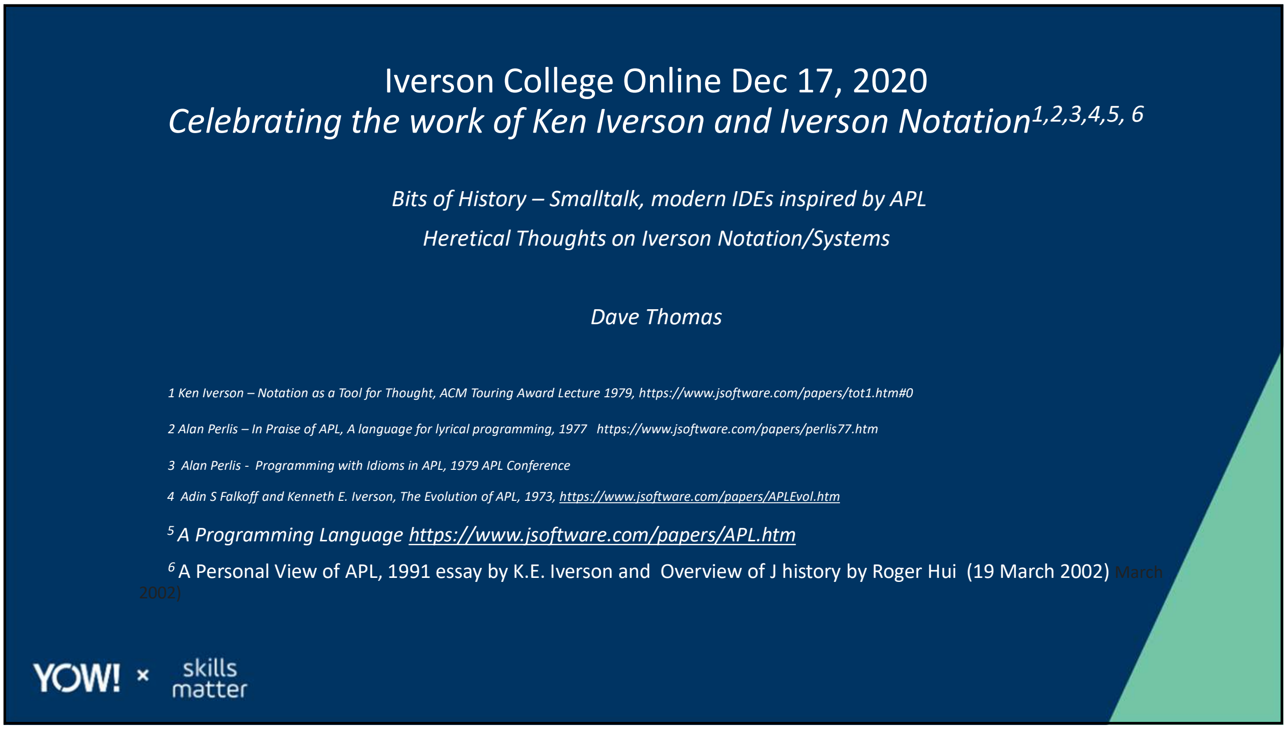


# Iverson College Online Dec 17, 2020
## *Celebrating the work of Ken Iverson and Iverson Notation*[1,2,3,4,5, 6]

*Bits of History – Smalltalk, modern IDEs inspired by APL*

*Heretical Thoughts on Iverson Notation/Systems*

*Dave Thomas*

*1 Ken Iverson – Notation as a Tool for Thought, ACM Touring Award Lecture 1979, https://www.jsoftware.com/papers/tot1.htm#0*

*2 Alan Perlis – In Praise of APL, A language for lyrical programming, 1977   https://www.jsoftware.com/papers/perlis77.htm*

*3  Alan Perlis -  Programming with Idioms in APL, 1979 APL Conference*

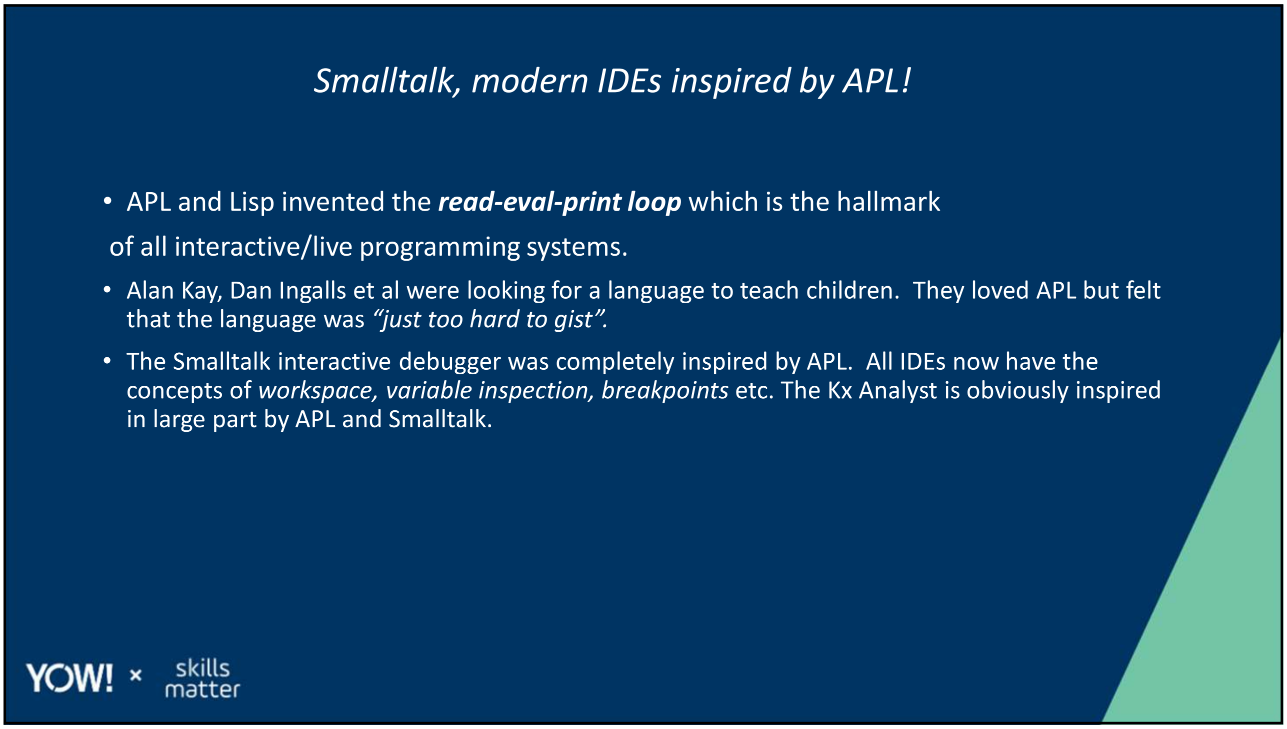
*4  Adin S Falkoff and Kenneth E. Iverson, The Evolution of APL, 1973, https://www.jsoftware.com/papers/APLEvol.htm*

[5] *A Programming Language https://www.jsoftware.com/papers/APL.htm*

[6] A Personal View of APL, 1991 essay by K.E. Iverson and  Overview of J history by Roger Hui  (19 March 2002) March 2002)

YOW! × skills matter

1

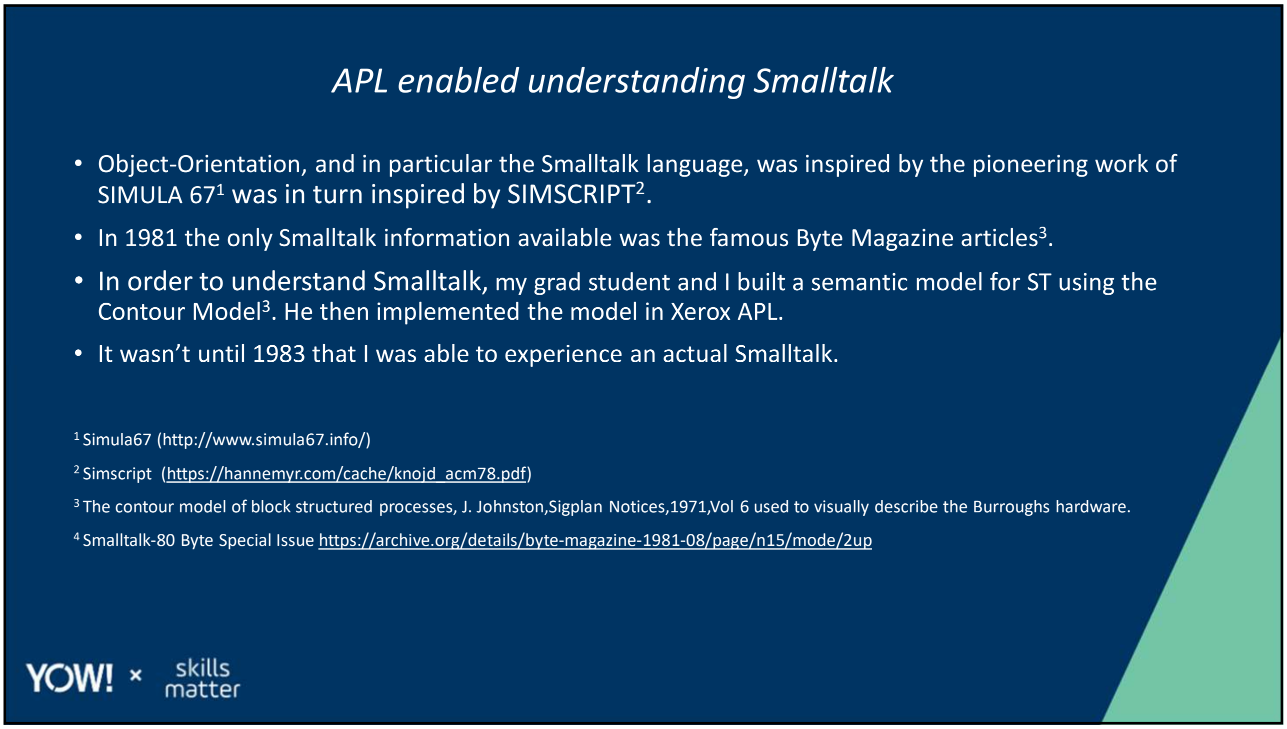


## *Smalltalk, modern IDEs inspired by APL!*

- APL and Lisp invented the ***read-eval-print loop*** which is the hallmark of all interactive/live programming systems.
- Alan Kay, Dan Ingalls et al were looking for a language to teach children.  They loved APL but felt that the language was *"just too hard to gist".*
- The Smalltalk interactive debugger was completely inspired by APL.  All IDEs now have the concepts of *workspace, variable inspection, breakpoints* etc. The Kx Analyst is obviously inspired in large part by APL and Smalltalk.

YOW! × skills matter

2



## *APL enabled understanding Smalltalk*

- Object-Orientation, and in particular the Smalltalk language, was inspired by the pioneering work of SIMULA 67[1] was in turn inspired by SIMSCRIPT[2].
- In 1981 the only Smalltalk information available was the famous Byte Magazine articles[3].
- In order to understand Smalltalk, my grad student and I built a semantic model for ST using the Contour Model[3]. He then implemented the model in Xerox APL.
- It wasn't until 1983 that I was able to experience an actual Smalltalk.

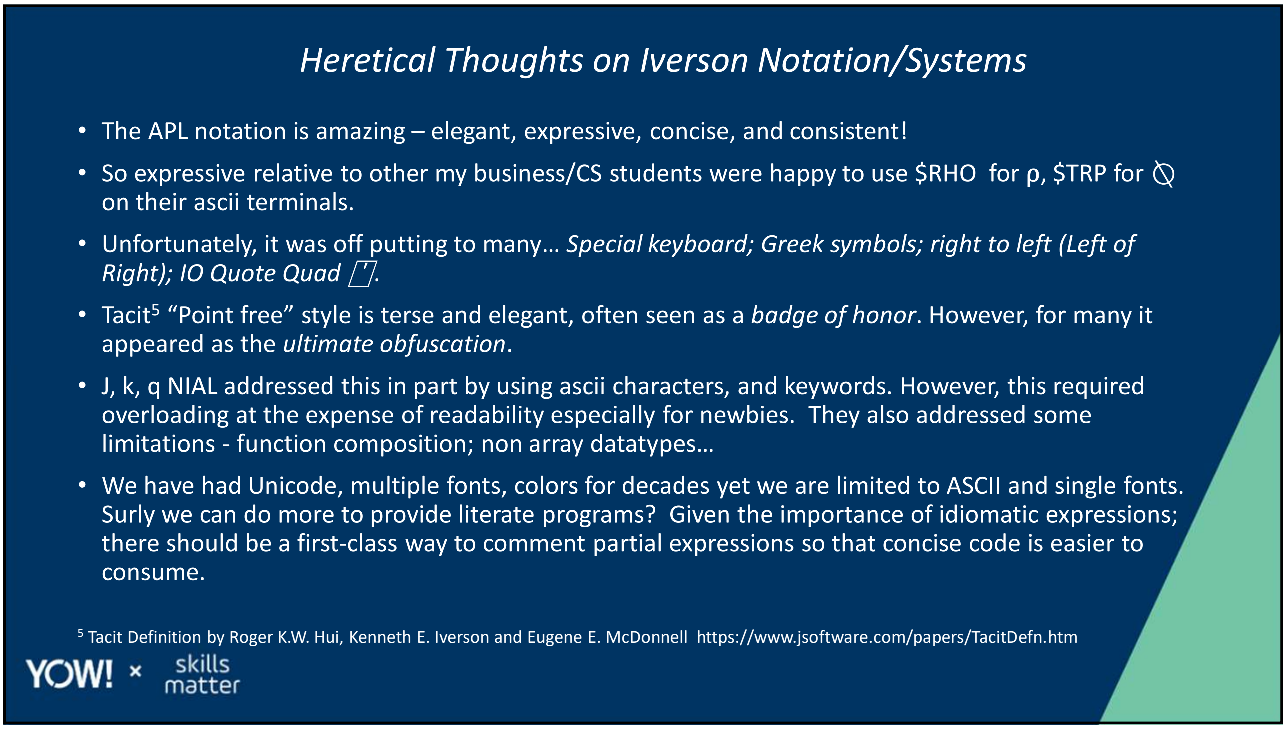
[1] Simula67 (http://www.simula67.info/)

[2] Simscript (https://hannemyr.com/cache/knojd_acm78.pdf)

[3] The contour model of block structured processes, J. Johnston,Sigplan Notices,1971,Vol 6 used to visually describe the Burroughs hardware.

[4] Smalltalk-80 Byte Special Issue https://archive.org/details/byte-magazine-1981-08/page/n15/mode/2up

YOW! × skills matter

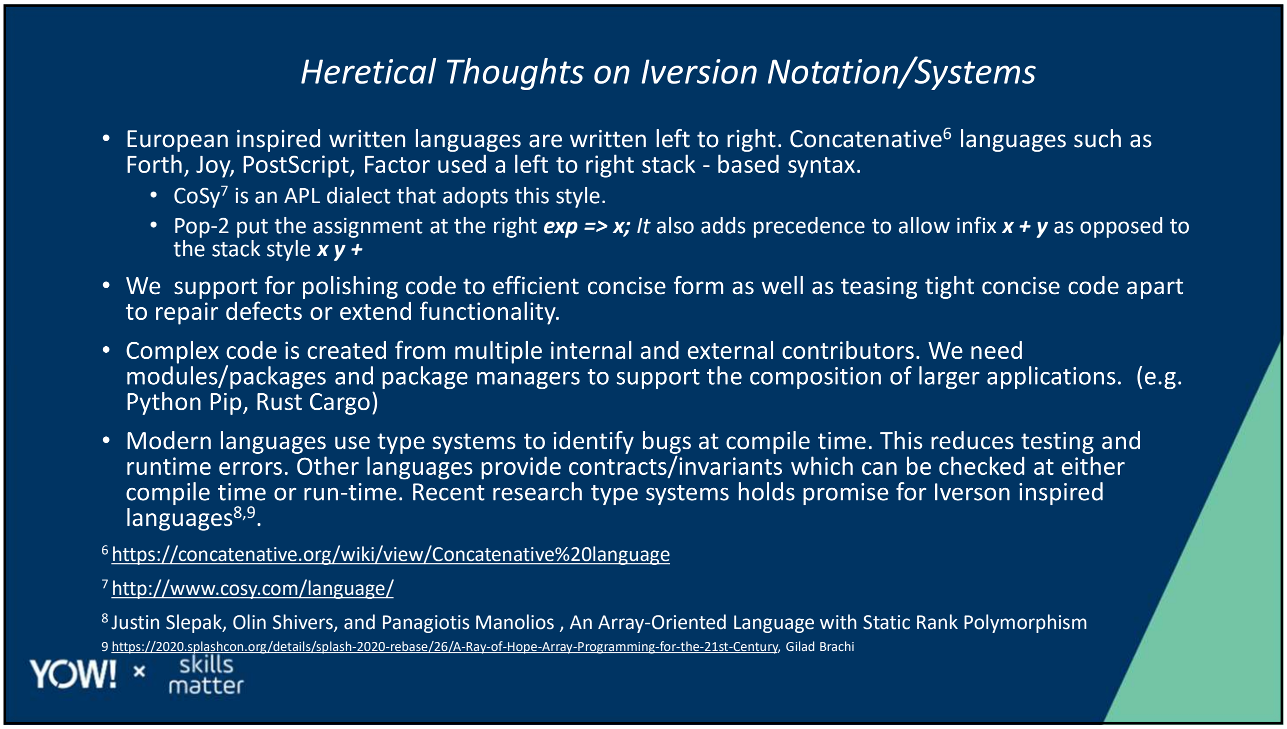


## *Heretical Thoughts on Iverson Notation/Systems*

- The APL notation is amazing – elegant, expressive, concise, and consistent!
- So expressive relative to other my business/CS students were happy to use $RHO for ρ, $TRP for ⍉ on their ascii terminals.
- Unfortunately, it was off putting to many… *Special keyboard; Greek symbols; right to left (Left of Right); IO Quote Quad ⍞.*
- Tacit[5] "Point free" style is terse and elegant, often seen as a *badge of honor*. However, for many it appeared as the *ultimate obfuscation*.
- J, k, q NIAL addressed this in part by using ascii characters, and keywords. However, this required overloading at the expense of readability especially for newbies. They also addressed some limitations - function composition; non array datatypes…
- We have had Unicode, multiple fonts, colors for decades yet we are limited to ASCII and single fonts. Surly we can do more to provide literate programs? Given the importance of idiomatic expressions; there should be a first-class way to comment partial expressions so that concise code is easier to consume.

[5] Tacit Definition by Roger K.W. Hui, Kenneth E. Iverson and Eugene E. McDonnell https://www.jsoftware.com/papers/TacitDefn.htm

YOW! × skills matter



## *Heretical Thoughts on Iversion Notation/Systems*

- European inspired written languages are written left to right. Concatenative[6] languages such as Forth, Joy, PostScript, Factor used a left to right stack - based syntax.
  - CoSy[7] is an APL dialect that adopts this style.
  - Pop-2 put the assignment at the right ***exp => x;*** *It* also adds precedence to allow infix ***x + y*** as opposed to the stack style ***x y +***
- We support for polishing code to efficient concise form as well as teasing tight concise code apart to repair defects or extend functionality.
- Complex code is created from multiple internal and external contributors. We need modules/packages and package managers to support the composition of larger applications. (e.g. Python Pip, Rust Cargo)
- Modern languages use type systems to identify bugs at compile time. This reduces testing and runtime errors. Other languages provide contracts/invariants which can be checked at either compile time or run-time. Recent research type systems holds promise for Iverson inspired languages[8,9].

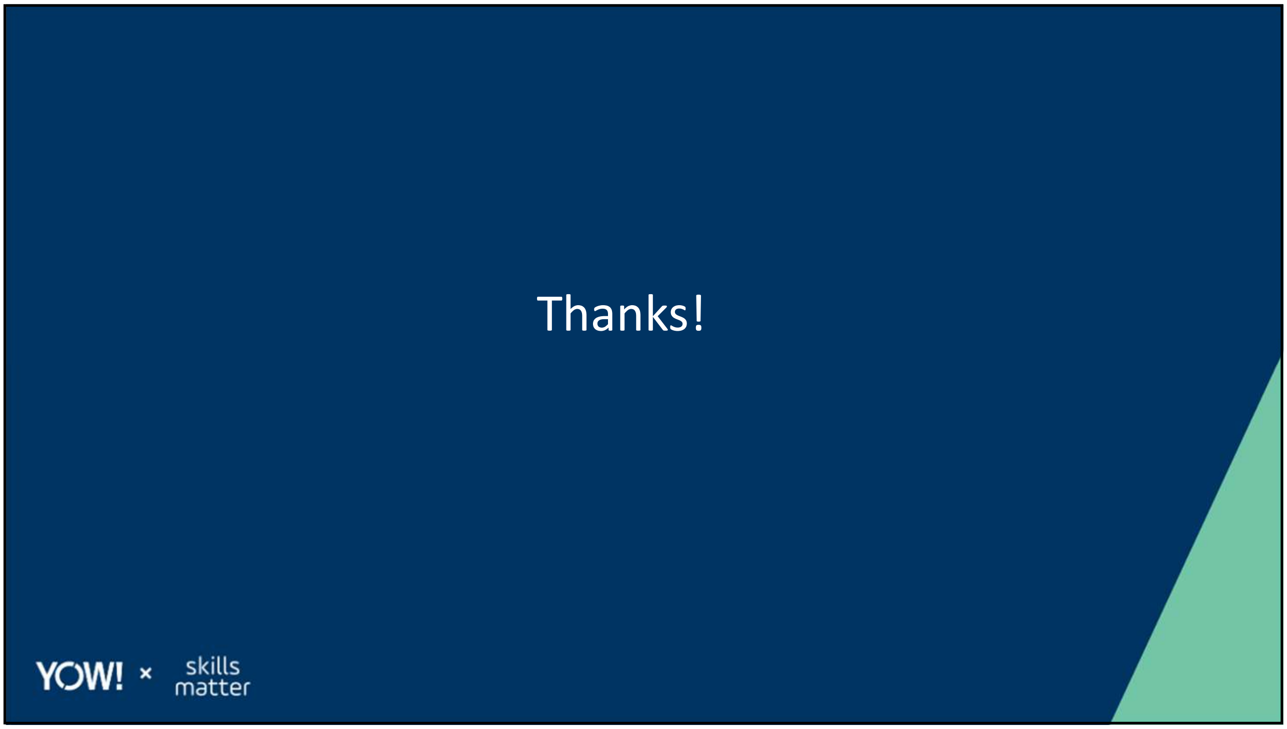
[6] https://concatenative.org/wiki/view/Concatenative%20language

[7] http://www.cosy.com/language/

[8] Justin Slepak, Olin Shivers, and Panagiotis Manolios , An Array-Oriented Language with Static Rank Polymorphism

[9] https://2020.splashcon.org/details/splash-2020-rebase/26/A-Ray-of-Hope-Array-Programming-for-the-21st-Century, Gilad Brachi

5



Thanks!

6