DYOL

Design a Language Know the Consequences

Garbage Collection

Automatic release of memory is impossible for cyclic data structures. Languages that want to support them, have a *garbage collector* — a runtime compiler component that occasionally *marks* data structures that have become inaccessible and then *sweeps* them away, freeing the memory. GC can compromise language responsiveness and performance.

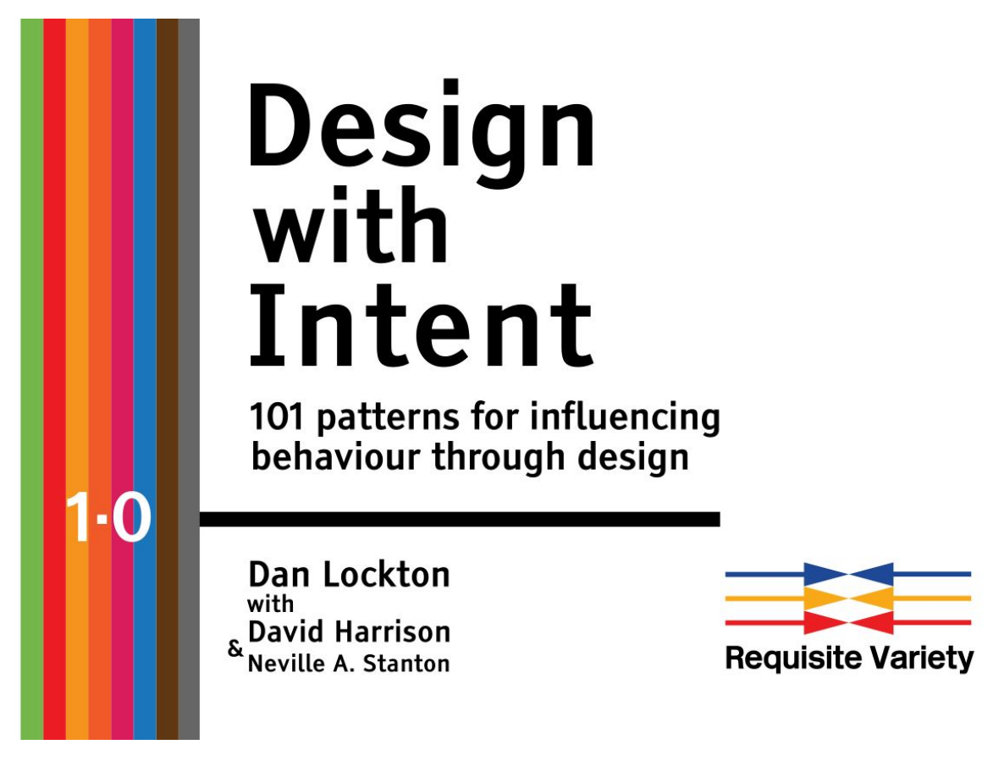
Dwl:Bundling, DB-RD:441, DB-PD:568, CC-DG:209, CD-GR:476, LI-PZ:471, PL-RS:117, PL-BM:443, LD-WH:123, SL-AS:444

Compilers
Principles, Techniques, and Tools
Alfred V. Aho
Ravi Sethi
Jeffrey D. Ullman
BRINCH HANSEN ON PASCAL COMPILERS
Modern Compiler Design
Second Edition
Engineering Modeling Languages

Concrete Syntax

The way to describe the concrete representation of the programs. The concrete syntax is used by humans to read, write, create and understand sentences of the language. Usually the only languages that do not have concrete syntax are those intended for internal intermediate representation. Some languages have more than one.

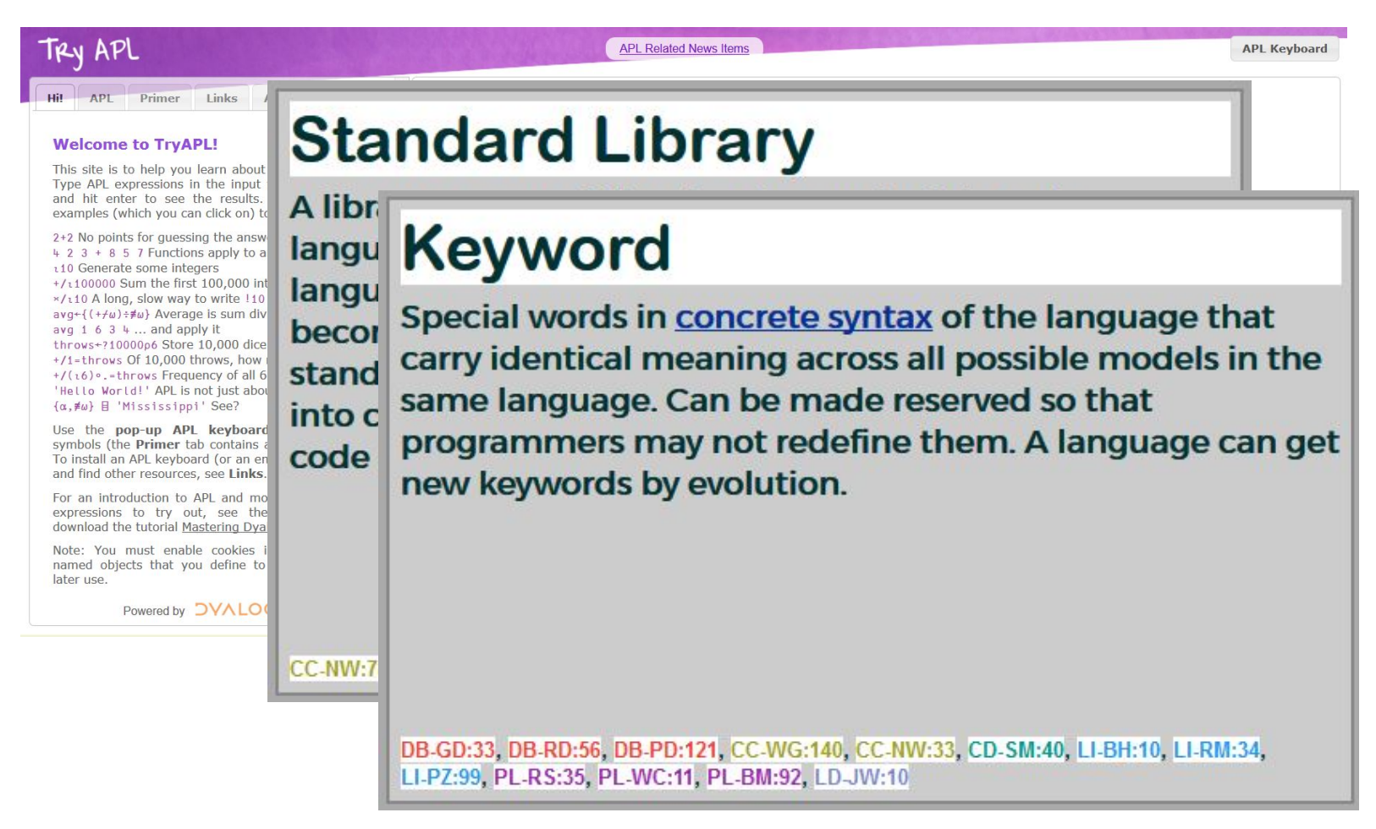
Dwl:Transparency, Wile:Artificial Language, Mernik:Notation, DB-GD:28, DB-RD:25, DB-PD:78, CC-WG:17, CD-AH:166, CD-GR:115, LI-BH:3, LI-PZ:41, PL-RS:124, PL-BM:89, PT-AO:39, PT-HU:2, PT-GJ:7, SL-CF:21, SL-RL:65



Alphabet

The basic alphabet is often taken for granted, especially for textual languages, but it is an important design aspect. In some languages (APL being the extreme) the alphabet is extremely broad, with specific symbols being used for built-in operators, which shifts the visual feel of the language closer to mathematics. In other languages keywords are taken from English, which limits language appeal to some groups of users (and may lead to reimplementations with translated keywords).

Dwl:Perceived affordances, DB-GD:28, DB-RD:92, DB-PD:165, CC-DG:15, CC-NW:10, CD-AH:52, LI-BH:10, PT-AO:34, PT-HU:1, PT-GJ:6, LD-ED:5



Standard Library

Keyword

Special words in concrete syntax of the language that carry identical meaning across all possible models in the same language. Can be made reserved so that programmers may not redefine them. A language can get new keywords by evolution.

DB-GD:33, DB-RD:56, DB-PD:121, CC-WG:140, CC-NW:33, CD-SM:40, LI-BH:10, LI-RM:34, LI-PZ:99, PL-RS:35, PL-WC:11, PL-BM:92, LD-JW:10

Operator Precedence

To avoid excessive use of parentheses, a language can provide a default convention of disambiguating constructs with 3+ entities bound by binary operators. In arithmetic expressions, the precedence usually follows mathematical laws.

DB-GD:47, DB-RD:31, DB-PD:86, CC-DG:103, CC-WG:28, CD-AH:819, CD-GR:158, LI-RM:71, LI-PZ:332, PL-RS:133, PL-WC:79, PL-BM:94, PT-GJ:266, LD-ED:9, LD-JW:30

Type Analysis §

Components can be identified, explicitly or automatically, to belong to a particular *type*. Among other things, the type determines applicability and compatibility of components with one another. In complex scenarios (like a monadic bind) hard to understand components can only fit together in one possible way. Type equivalence rules can be based on names, structure, scopes, etc.

Dwl:Matched affordances, Wile:Type Checking, DB-GD:49, DB-RD:343, DB-PD:56, CC-WG:26, CD-AH:489, CD-GR:521, LI-BH:110, LI-RM:91, LI-PZ:195, PL-RS:38, PL-BM:129, PT-AO:98, LD-WH:13, SL-RL:267

Related cards

- Character Type
- Class (main type in OOP)
- Collection (Composite Type)
- Enumeration Type
- Heterogeneous Data
- Numeric Data Type
- Parametrised Type
- Picture Clause
- Pointer (Reference Type)
- Record (and Dictionary)
- Scope & Binding (Name-Type Binding)
- Substitution (Subtyping)
- Type Definition (Used-defined Types)

Synonyms and similar terms

Type Checking
: Can be used as a synonym for type analysis, but also as an umbrella term for type analysis and synthesis: all rules and actions around the type systems.

Type Synthesis
: A complementary set of techniques to type analysis, used in software language implementations. The main difference is the direction of type computations: bottom up in synthesis and top down in analysis.

Type System
: The set of rules combining all the types available in a software language, into one system with subtypes, conversions, etc.

Full deck at http://slebok.github.io/dyol

Vadim Zaytsev aka @grammarware

raincode LABS compiler experts