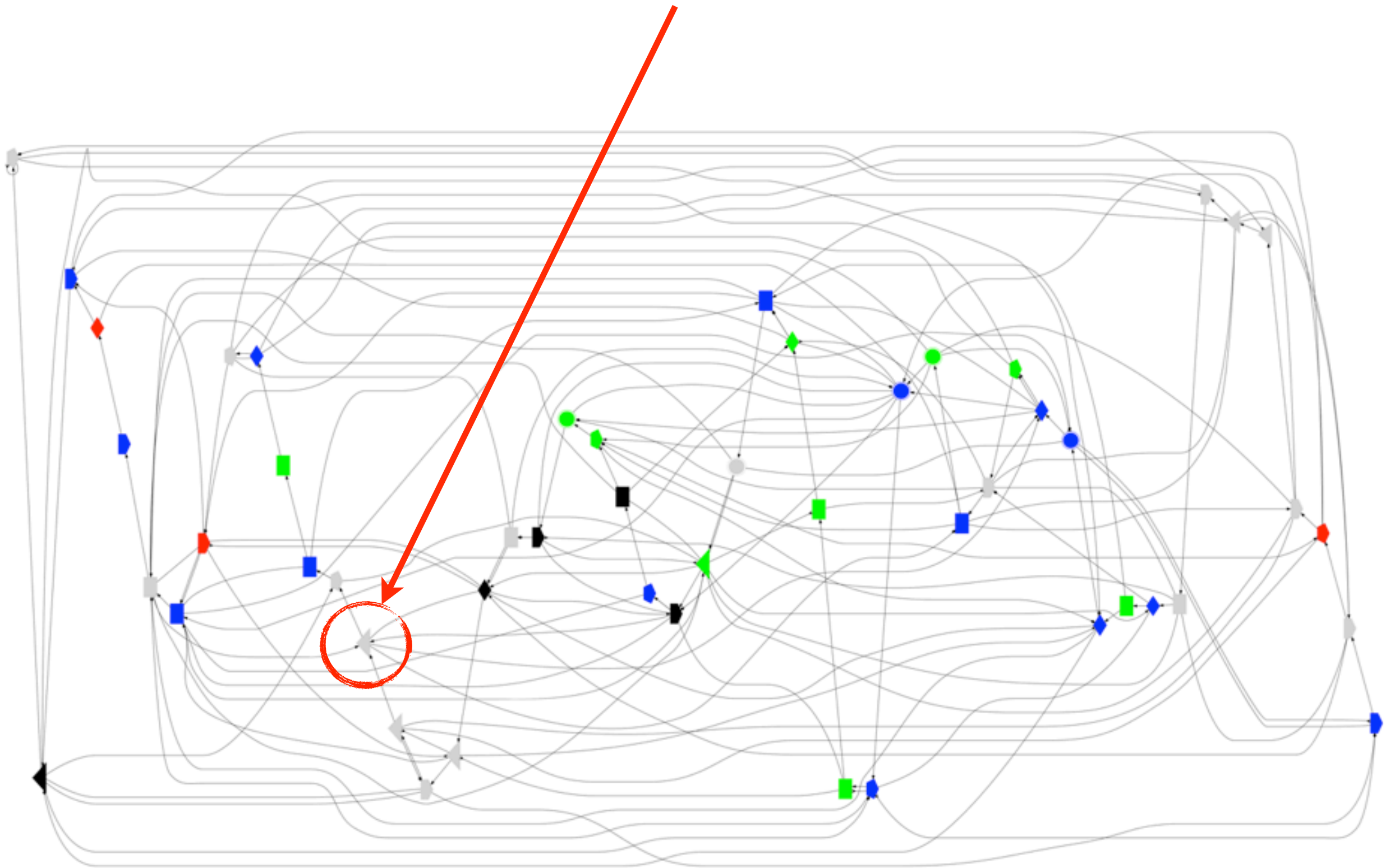


edn and Fressian: Languages of the System

@stuarthalloway

You Are Here



Narcissistic Design

State Transition Model

Processing Model

Topology

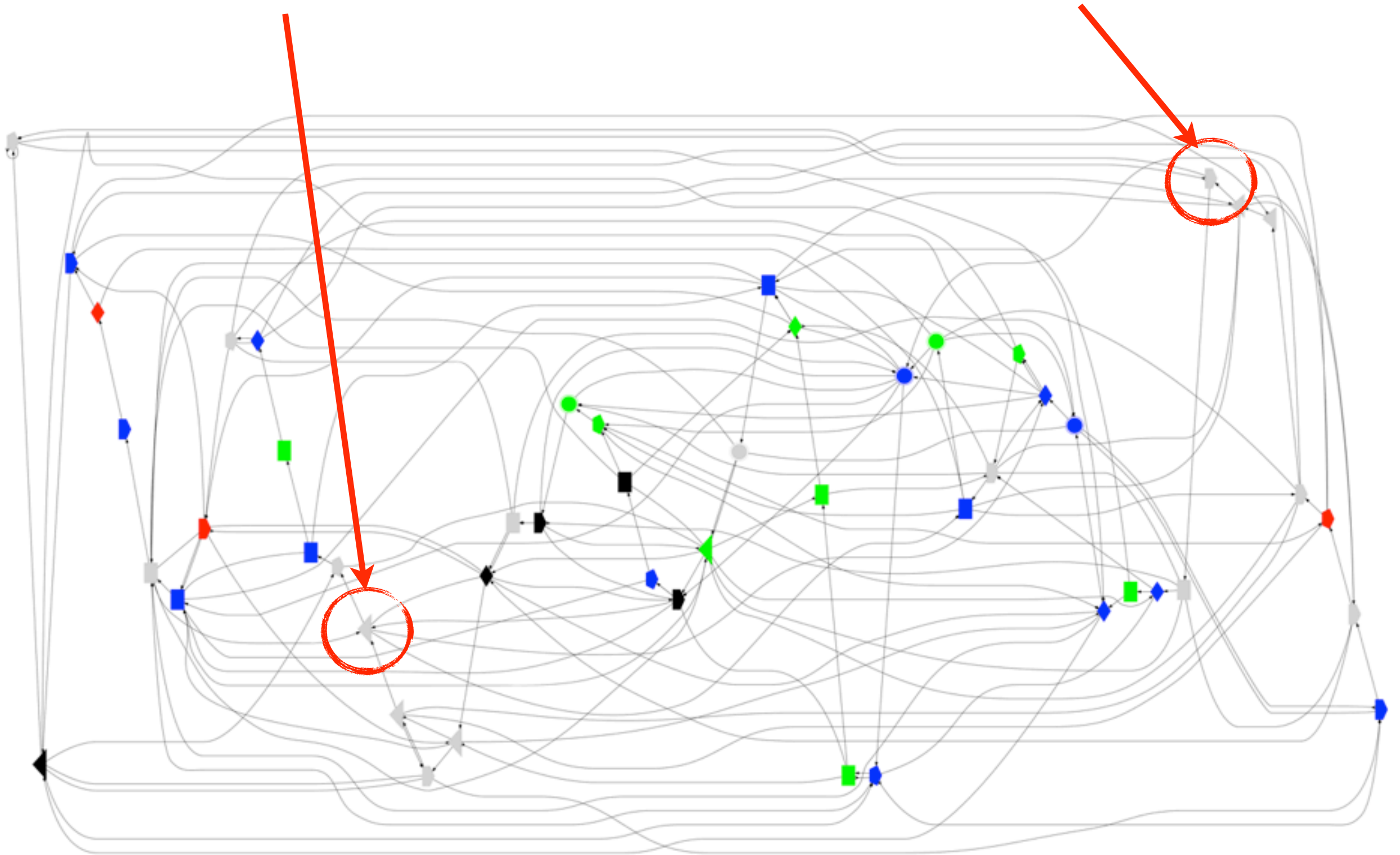
Availability

Static Typing

Centralized Authority

Language *X* (*for all values of X!*)

Change Here ... Impact Here?



Leverage

Structure

Extensibility

Composability

Universal Data

Self-describing

Immutable Values

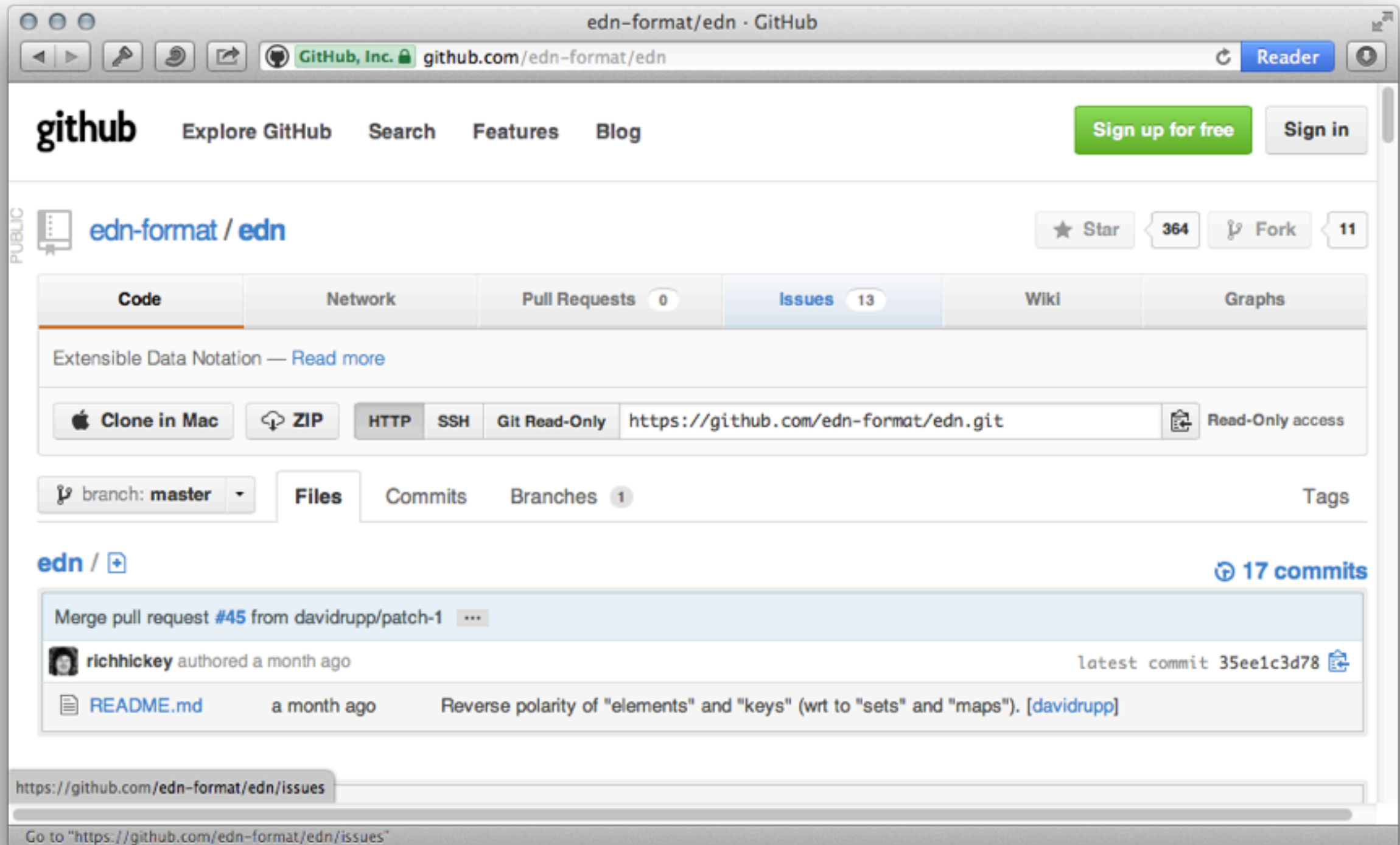
Namespaces

Extensible

Composable

Batteries-included

edn-format.org



edn Example

```
{ :firstName "John"  
  :lastName "Smith"  
  :age 25  
  :address {  
    :streetAddress "21 2nd Street"  
    :city "New York"  
    :state "NY"  
    :postalCode "10021" }  
  :phoneNumber  
    [ { :type "name" :number "212 555-1234"}  
      { :type "fax" :number "646 555-4567" } ] }
```


edn is

Subset of Clojure syntax

Used by Datomic and others as data transfer format

Language/implementation neutral

Open

edn Is NOT

Type System

Schema Based

Object-Oriented

Scalars 1

<code>nil</code>	nil, null, or nothing
booleans	<code>true</code> or <code>false</code>
strings	enclosed in “ <code>double quotes</code> ” may span multiple lines <code>\t \r \n</code> supported
characters	<code>\c</code> <code>\newline, \return, \space</code> and <code>\tab</code>

Scalars 2: Numbers

integers

0-9
negative

floating point

64-bit (double) precision is expected.

Scalars 3: Names

symbols	<p>used to represent identifiers should map to something other than strings may include <i>namespace</i> prefixes: <code>my-namespace/foo</code></p>
keywords	<p>identifiers that designate themselves semantically akin to enumeration values symbols that must start with : <code>:fred</code> or <code>:my/fred</code></p>

Collections 1

lists

a sequence of values
zero or more elements within `()`
`(a b 42)`

vectors

a sequence of values...
...that supports random access
zero or more elements within `[]`
`[a b 42]`

Collections 2

maps

collection of key/value associations
every key should appear only once
unordered

zero or more elements within `{ }`
`{ :a 1, "foo" :bar, [1 2 3] four }`

sets

collection of unique values
unordered
heterogeneous
zero or more elements within `#{ }`
`#{a b [1 2 3]}`

Disregard

comments

;

discard

#_ is the discard sequence
read & discard the next element
`[a b #_foo 42] => [a b 42]`

Value Equality

nil, booleans, strings, characters, and symbols are equal to values of the same type with the same edn representation.

integers and floating point numbers should be considered equal to values only of the same magnitude, type, and precision.

sequences (lists and vectors) are equal to other sequences whose count of elements is the same, and for which each corresponding pair of elements (by ordinal) is equal

Value Equality

sets are equal if they have the same count of elements and, for every element in one set, an equal element is in the other.

maps are equal if they have the same number of entries, and for every key/value entry in one map an equal key is present and mapped to an equal value in the other.

tagged elements must define their own equality semantics.

Extensibility: Tagged Elements

#name edn-form

Name describes interpretation of following element

Recursively defined

Built-in Tagged Elements

#inst "rfc-3339-format"

tagged element is a string in RFC-3339 format

#uuid "f81d4fae-7dec-11d0-a765-00a0c91e6bf6"

tagged element is a canonical UUID string

Implementing a Tag Handler

associate a qualified name

```
Parser.Config cfg =  
    Parsers.newParserConfigBuilder()  
        .putTagHandler(Tag.newTag("us.bpsm", "uri"),  
            new TagHandler() {  
                public Object transform(Tag tag, Object value) {  
                    return URI.create((String) value);  
                }  
            })  
        .build();  
Parser p = Parsers.newParser(cfg);  
Parseable pbr = Parsers.newParseable(p, "  
    \"#us.bpsm/uri \\\"http://example.com\\\"\" );  
assertEquals(new URI("http://example.com"), p.nextValue(pbr));
```

with an interpretation

<https://github.com/bpsm/edn-java>

Fressian: Lean & Mean

Still no narcissism!

Fressian

The screenshot shows a web browser window with the GitHub repository page for `Datomic/fressian`. The browser's address bar shows the URL `https://github.com/Datomic/fressian/wiki/Rationale`. The GitHub header includes the logo, a search bar, and navigation links like `Explore`, `Features`, `Enterprise`, `Blog`, `Sign up`, and `Sign in`. The repository name `Datomic / fressian` is displayed with `77` stars and `5` forks. Below this is a navigation bar with tabs for `Code`, `Network`, `Pull Requests` (1), `Issues` (4), `Wiki` (selected), and `Graphs`. Under the `Wiki` tab, there are links for `Home`, `Pages`, `Wiki History`, and `Git Access`. The main content area is titled `Rationale` with a `Page History` button. The text states: "Fressian is designed to communicate **values** in a way that is:" followed by a bulleted list of features:

- self-describing
- schema-free
- extensible
- batteries-included
- binary
- efficient

 The final sentence reads: "Each of these is elaborated below."

Rationale

Fressian is designed to communicate **values** in a way that is:

- self-describing
- schema-free
- extensible
- batteries-included
- binary
- efficient

Each of these is elaborated below.

Fressian is Similar to edn

Self-describing

Immutable Values

Namespaces

Extensible

Composable

Batteries-included

Fressian is Also

Binary

Byte-code driven

Primitive aware

Domain compressible

Basics

```
public Writer writeObject(Object o);  
public Writer writeAs(String tag, Object o);  
  
public Object readObject();
```


Primitives

```
public Writer writeBoolean(boolean b) throws IOException;
public Writer writeBoolean(Object o) throws IOException;
public Writer writeInt(long l) throws IOException;
public Writer writeInt(Object o) throws IOException;
public Writer writeDouble(double d) throws IOException;
public Writer writeDouble(Object o) throws IOException;
public Writer writeFloat(float d) throws IOException;
public Writer writeFloat(Object o) throws IOException;
public Writer writeString(Object o) throws IOException;
public Writer writeList(Object l) throws IOException;
public Writer writeBytes(byte[] b) throws IOException;
public Writer writeBytes(byte[] b, int offset, int length);
```

Understanding is Optional


```
public interface Tagged {  
    public Object getTag();  
    public Object getValue();  
    public Map getMeta();  
}
```

readers can return Tagged
in lieu of any concrete
struct or object



```
public Writer writeTag(Object tag, int componentCount);
```

writers can dynamically
create tagged types with
no structure in hand




Caching

tell writer to cache...



```
public Writer writeObject(Object o, boolean cache);  
public Writer writeAs(String tag, Object o, boolean cache);  
  
public interface Cached {  
    public Object getObjectToCache();  
}
```



...or object can know it
wants caching

Caching Example

```
new WriteHandler() {  
    public void write(Writer w, Object instance) throws IOException {  
        w.writeTag(tag, 6);  
        Datom d = (Datom) instance;  
        // elided write five other tags  
        w.writeObject(d.getT(), true);  
    }  
};
```

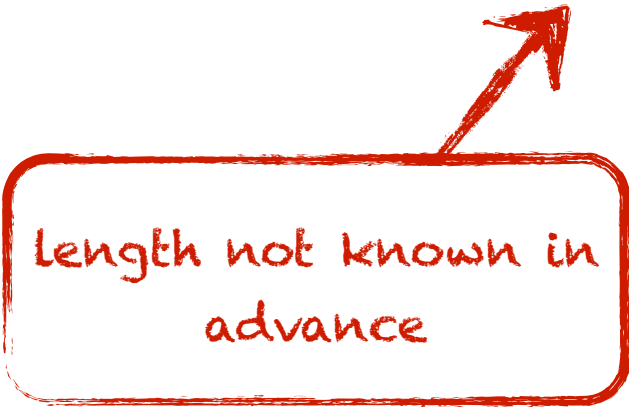


Datomic

Datomic efficiently stores the time T of every atomic fact in the system

Streaming

```
public Writer beginClosedList() throws IOException;  
public Writer endList() throws IOException;
```



length not known in
advance



manage ongoing
cache state

```
writer.resetCaches();
```

Packed Representations

Encode type and part|all of a value in a single byte

0x00 means “numeric” and “value 0”

0xd1 mean “byte array” and “length 1”

Chunked Representations

Work in constrained memory, nothing $> 64k$

0xE2 precedes nonterminal chunk of a string

0xE3 precedes terminal chunk of a string

Similar for bytes

Byte Code Overview

0 - 0x40	<i>themselves</i>
0x40 - 0x80	<i>cascading packed integers</i>
0xF8	<i>a.p. integer</i>
0xD0 - 0xD8	<i>packed byte arrays</i>
0x80 - 0xA0	<i>single-byte cache codes</i>
0xCC, 0xCD	<i>get/put cache</i>
0xC5	<i>URI</i>

<https://github.com/Datomic/fressian/wiki/Bytecodes>

<https://github.com/Datomic/fressian/blob/master/test/org/fressian/codegen.clj>

edn and Fressian are

Value formats

Self-describing

Extensible

Schemaless

Designed for an decentralized world

Languages of the System

Resources

The Language of the System

https://www.youtube.com/watch?v=ROor6_NGIWU

edn

<http://edn-format.org>. The edn specification.

<http://clojure.com>. The Clojure language.

<https://github.com/edn-format/edn/wiki/Implementations>. edn implementations,

Fressian

<https://github.com/Datomic/fressian>. The Fressian spec and reference impl.

<http://www.datomic.com/>. Datomic.

<https://github.com/fressian/fressian-clr>. Fressian on the CLR.

Stuart Halloway

<https://github.com/stuarthalloway/presentations/wiki>. Presentations

<http://thinkrelevance.com/blog/tags/podcast>. The Relevance Podcast.

<http://www.linkedin.com/pub/stu-halloway/0/110/543/>

<https://twitter.com/stuarthalloway>