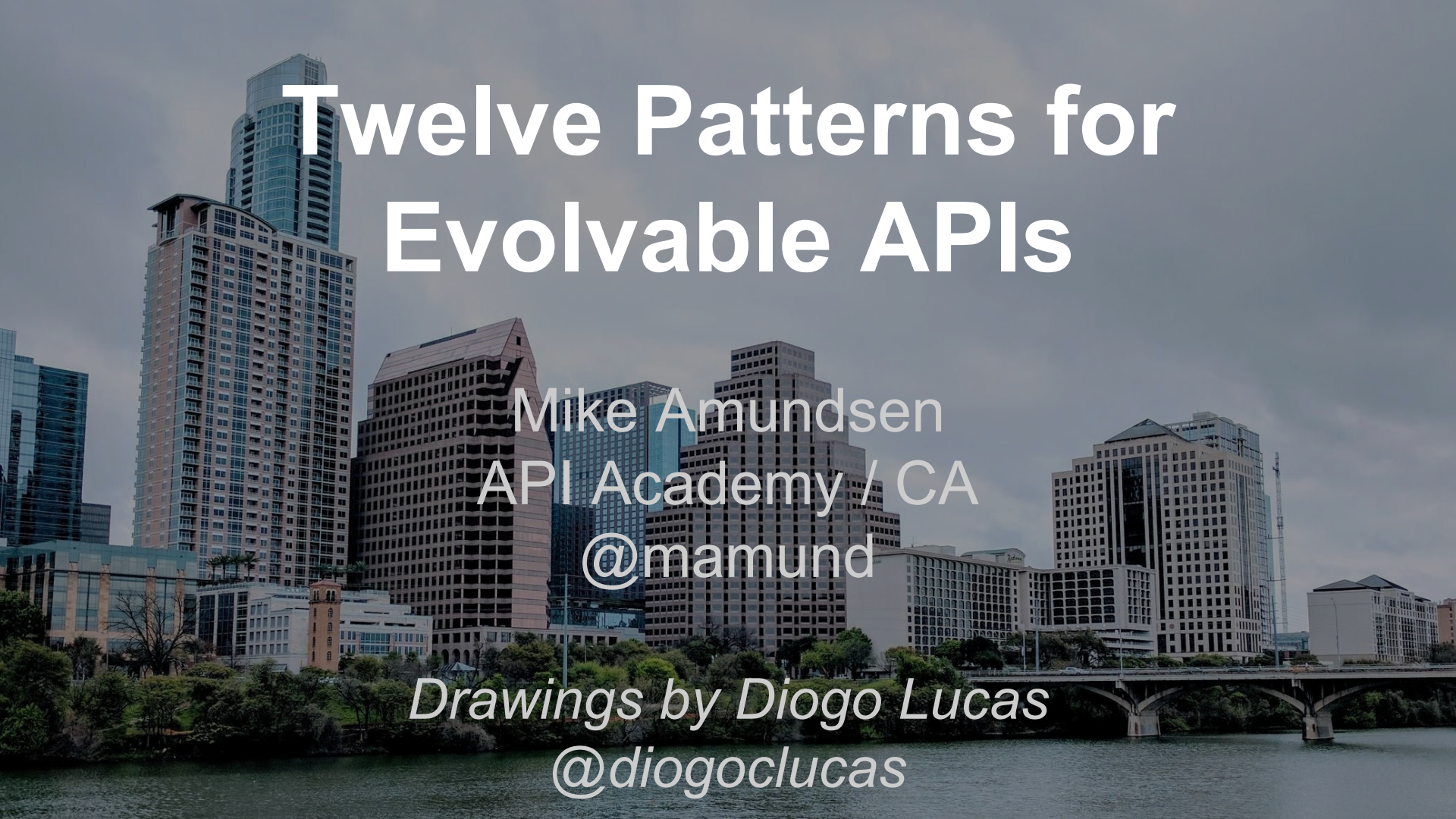


Twelve Patterns for Evolvable APIs

Mike Amundsen
API Academy / CA
@mamund

Drawings by Diogo Lucas
@diogoclucas

Introduction


Ctrl
PgUp
Mike Amundsen
@mamund

http://apiacademy.co

Creating Evolvable Hypermedia Applications

Building

Hypermedia APIs with HTML5 & No

O'REILLY®

Mike

O'REILLY®

Designing APIs for the Web

Mike Amundsen

VIDEO

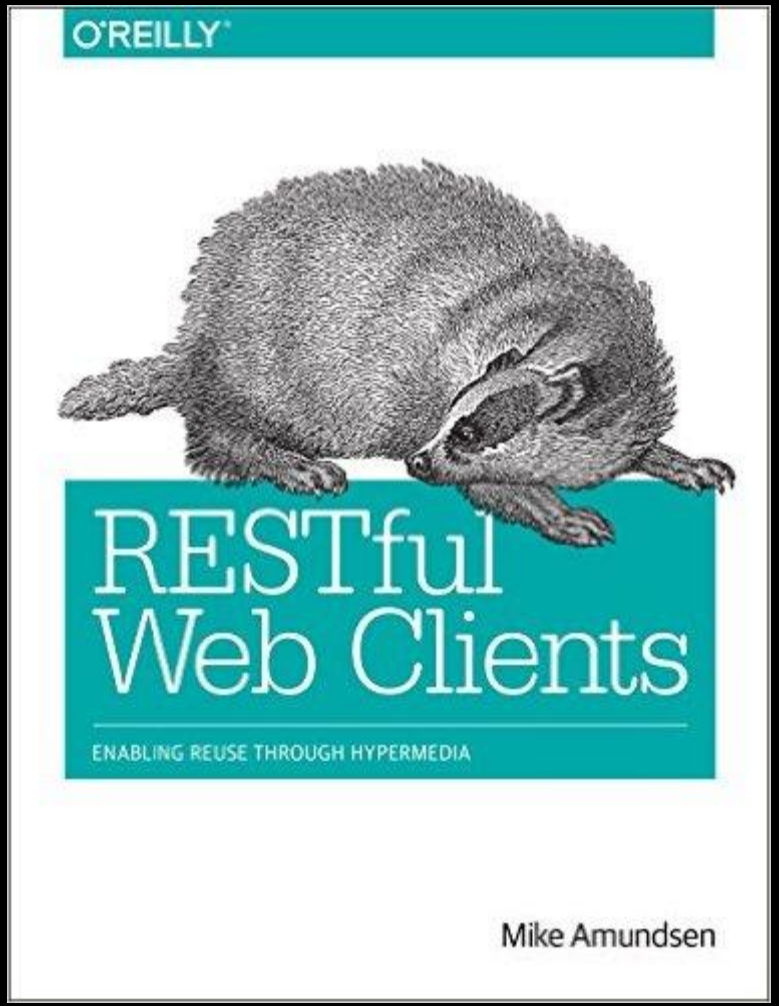
Services for a Changing World

RESTful Web APIs

O'REILLY®

Leonard Richardson, Mike Amundsen & Sam Ruby

@RWCBook


O'REILLY
RESTful Web Clients
ENABLING REUSE THROUGH HYPERMEDIA
Mike Amundsen

Outline

- Hypermedia
- Messages
- Patterns
- 4+4+4
- Summary

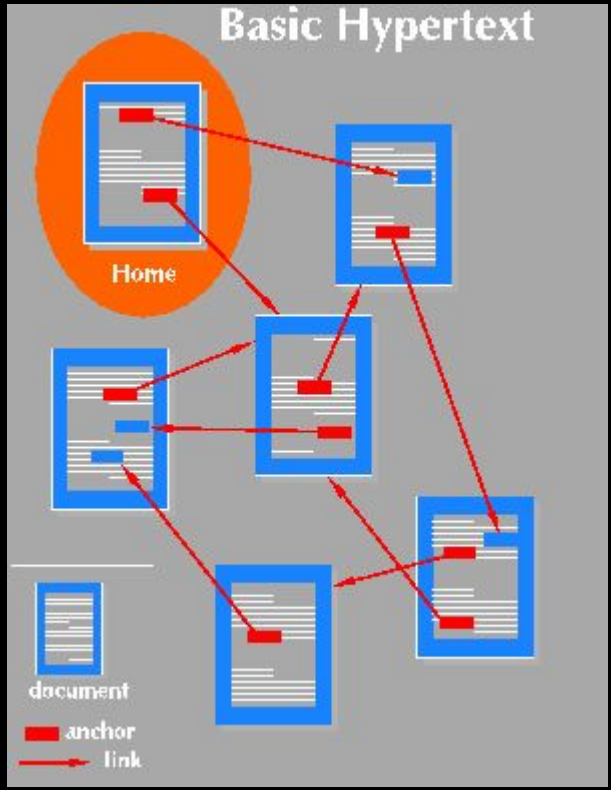
Hypermedia

What is Hypermedia?

Hypertext is text which is not constrained to be linear. Hypertext is text which contains links to other texts.

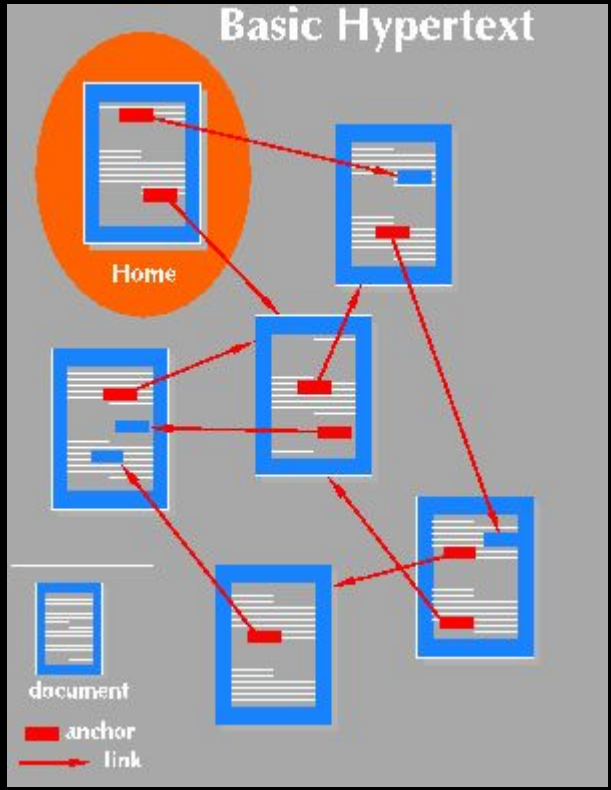
The term was coined by Ted Nelson around 1965.

https://www.w3.org/WhatIs.html



What is Hypermedia?

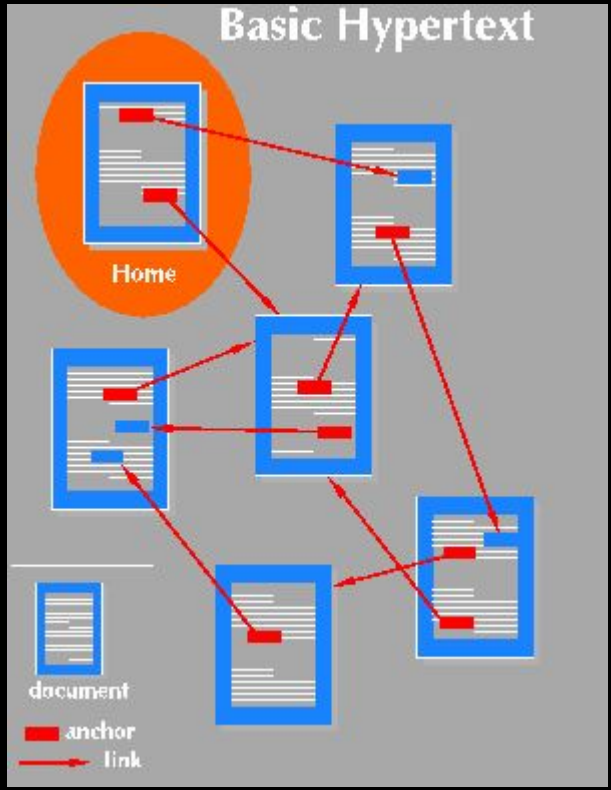
Hypermedia is a term used for hypertext which is not constrained to be text: it can include graphics, video and sound, for example.



https://www.w3.org/WhatIs.html

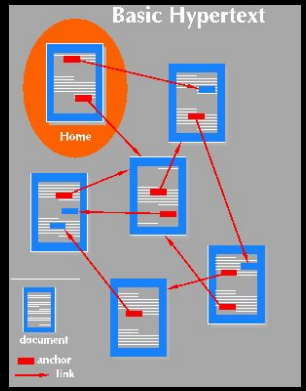
What is Hypermedia?

Hypertext and Hypermedia are concepts, not products.



https://www.w3.org/WhatIs.html

Hypermedia is the language of the WWW



Why Hypermedia?

Why Hypermedia?
Affordances!

Affordances

"The affordances of the environment are what it offers ... what it provides or furnishes, either for good or ill.

James Gibson, 1977

Affordances

“The term affordance refers to the perceived and actual properties of the thing, primarily those fundamental properties that determine just how the thing could possibly be used.”

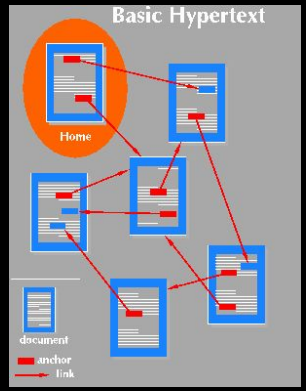
Donald Norman, 1988

Affordances

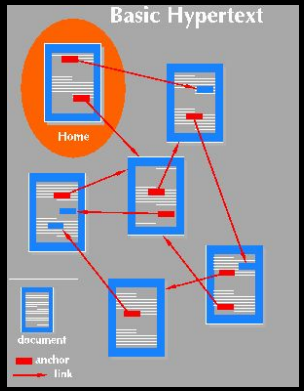
“When I say Hypertext, I mean the simultaneous presentation of information and controls such that the information becomes the affordance through which the user obtains choices and selects actions.”

Roy Fielding, 2008

Affordances are the reason for hypermedia



So, what does it look like?



HTML

```
1  <title>Hypertext Links</title>
2  <h1>Links and Anchors</h1>
3  A link is the connection between one piece of
4  <a href=WhatIs.html>hypertext</a> and another.
5
```

https://www.w3.org/History/19921103-hypertext/hypertext/WWW/Link.html

Atom

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">

  <title>Example Feed</title>
  <link href="http://example.org/"/>
  <updated>2003-12-13T18:30:02Z</updated>
  <author>
    <name>John Doe</name>
  </author>
  <id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>

  <entry>
    <title>Atom-Powered Robots Run Amok</title>
    <link href="http://example.org/2003/12/13/atom03"/>
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
```

https://tools.ietf.org/html/rfc4287

CCXML

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <ccxml version="1.0" xmlns="http://www.w3.org/2002/09/ccxml">
3   <eventprocessor>
4     <transition event="connection.alerting">
5       <var name="MyVariable" expr="'This is a CCXML Variable'"/>
6       <log expr="'Hello World. I just made a variable: ' + MyVariable"/>
7       <log expr="'Lets hang up on this incoming call.'"/>
8       <exit/>
9     </transition>
10  </eventprocessor>
11 </ccxml>
```

https://www.w3.org/TR/ccxml/

HAL

```
{
    "_links": {
        "self": { "href": "/orders" },
        "curies": [{ "name": "ea", "href": "http://example.com/docs/rels/{rel}",
        "next": { "href": "/orders?page=2" },
        "ea:find": {
            "href": "/orders{?id}",
            "templated": true
        },
        "ea:admin": [{
            "href": "/admins/2",
            "title": "Fred"
        }, {
            "href": "/admins/5",
            "title": "Kate"
        }]
    },
    "currentlyProcessing": 14,
    "shippedToday": 20,
    "_embedded": {
```

https://www.w3.org/TR/ccxml/

Siren

```
"actions": [
  {
    "name": "add-item",
    "title": "Add Item",
    "method": "POST",
    "href": "http://api.x.io/orders/42/items",
    "type": "application/x-www-form-urlencoded",
    "fields": [
      { "name": "orderNumber", "type": "hidden", "value": "42" },
      { "name": "productCode", "type": "text" },
      { "name": "quantity", "type": "number" }
    ]
  }
],
"links": [
  { "rel": [ "self" ], "href": "http://api.x.io/orders/42" },
```

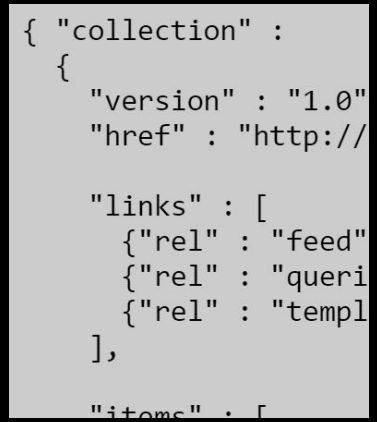
https://github.com/kevinswiber/siren

Collection+JSON

```
{ "collection" :
  {
    "version" : "1.0",
    "href" : "http://example.org/friends/",

    "links" : [
      {"rel" : "feed", "href" : "http://example.org/friends/rss"},
      {"rel" : "queries", "href" : "http://example.org/friends/?queries"},
      {"rel" : "template", "href" : "http://example.org/friends/?template"}
    ],

    "items" : [
      {
        "href" : "http://example.org/friends/jdoe",
        "data" : [
          {"name" : "full-name", "value" : "J. Doe", "prompt" : "Full Name"},
          {"name" : "email", "value" : "jdoe@example.org", "prompt" : "Email"}
        ],
        "links" : [
          {"rel" : "blog", "href" : "http://examples.org/blogs/jdoe", "prompt" : "Bl
          {"rel" : "avatar", "href" : "http://examples.org/images/jdoe", "prompt" :
        ]
      }
```

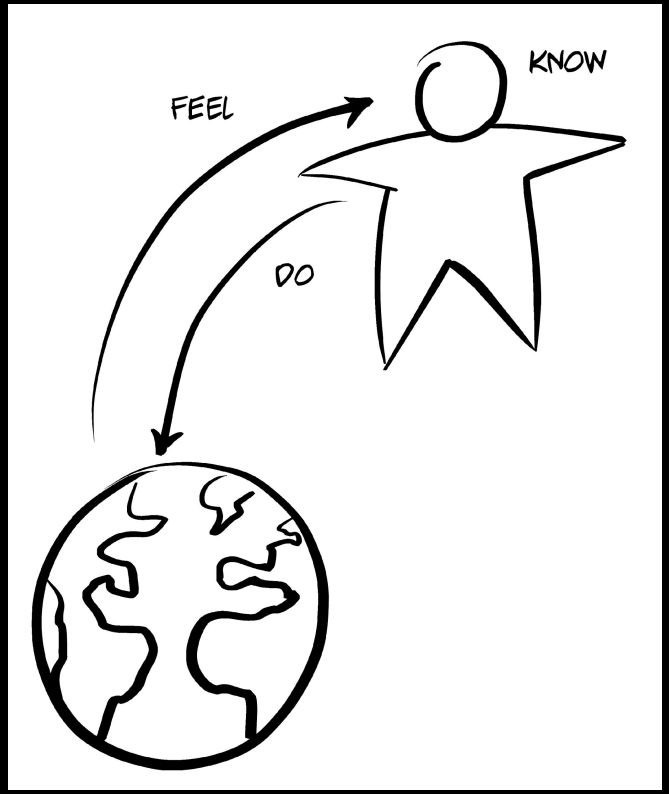
http://amundsen.com/media-types/collection/

Hypermedia Types are the programming language of the WWW



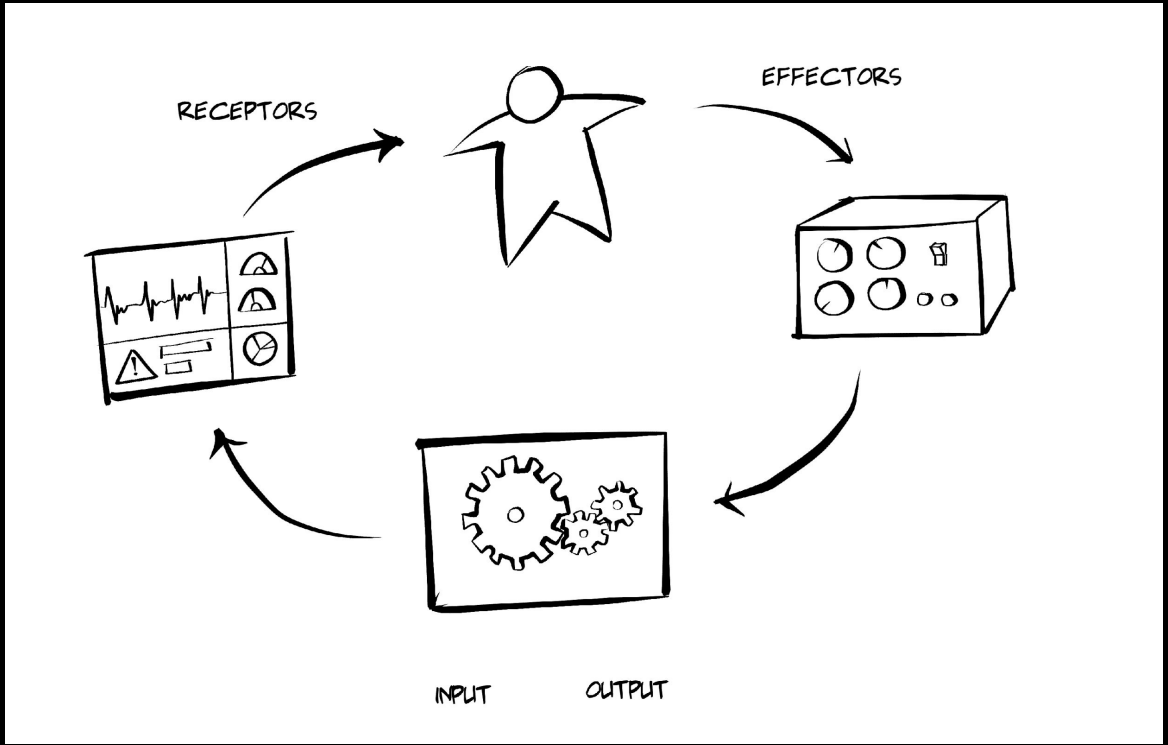
Messages

Messages are how we communicate



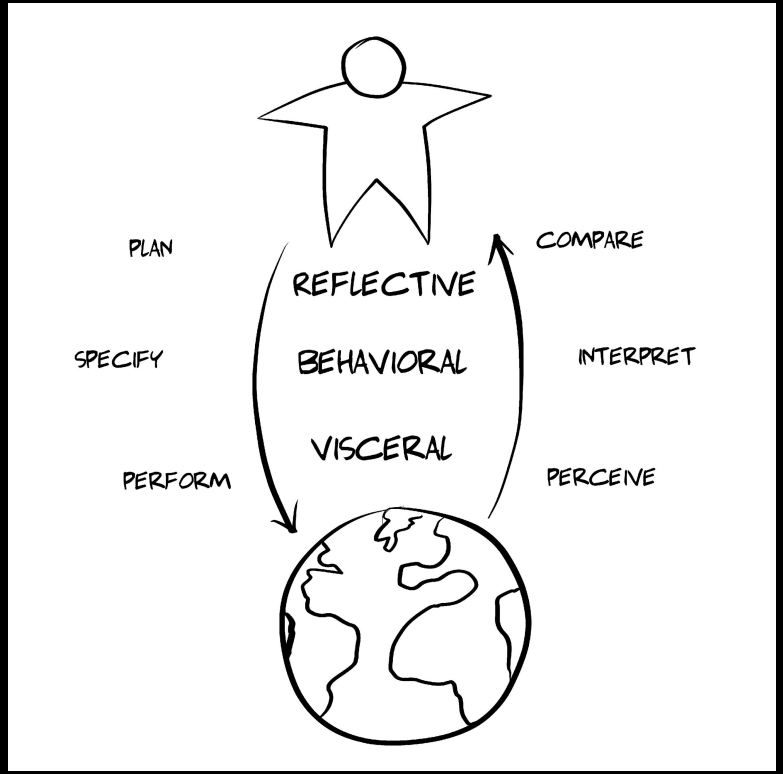
Bill Verplank

Messages are how we manipulate



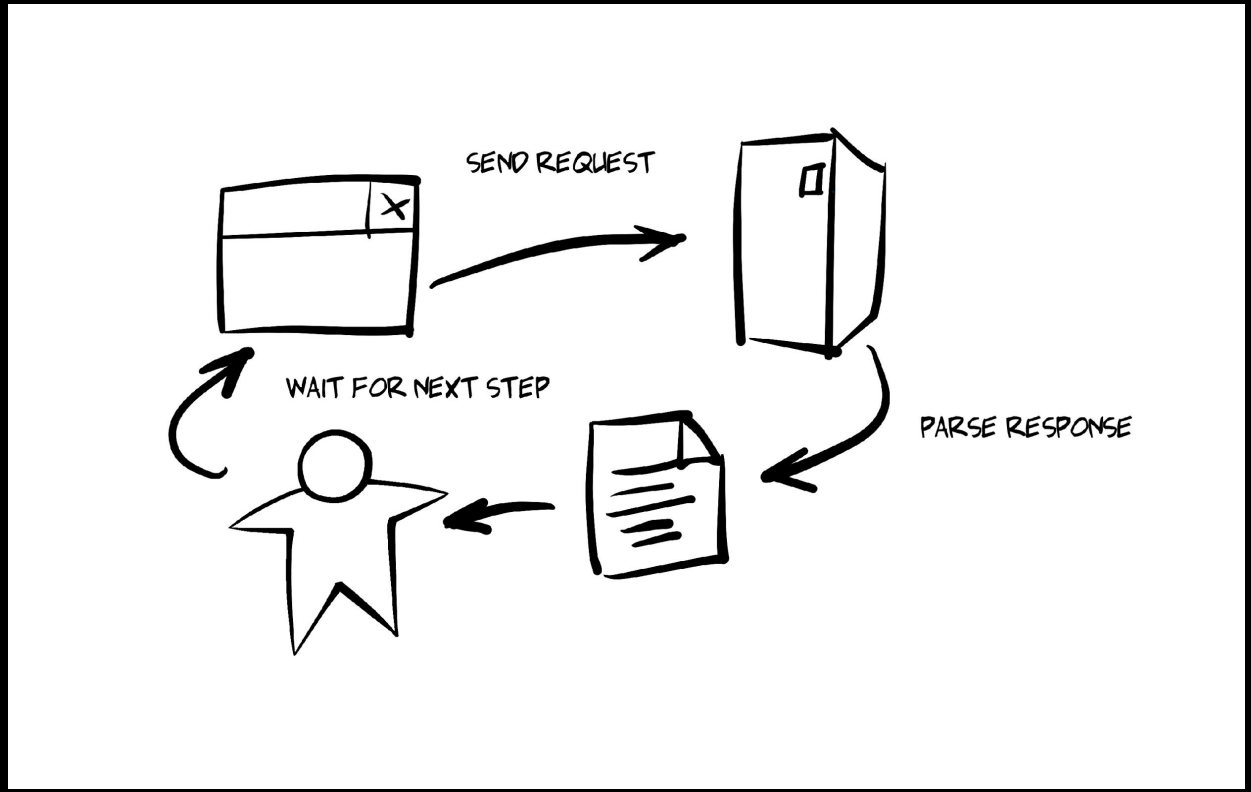
The Ulm Design School

We manipulate via affordances



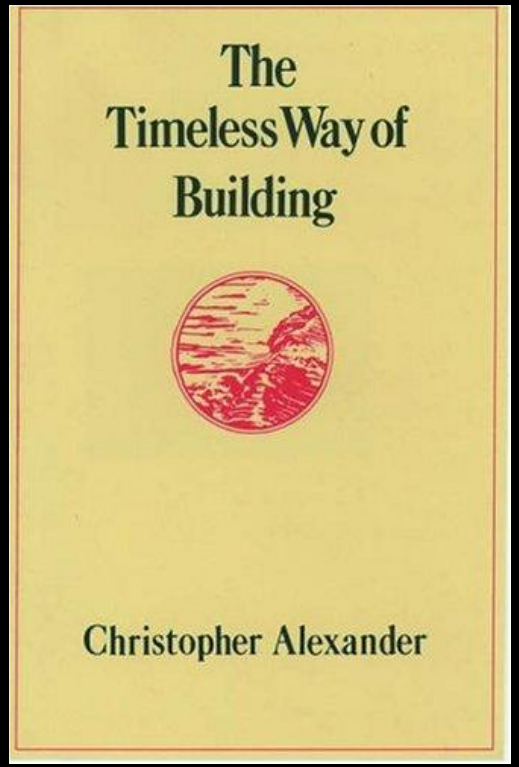
Donald Norman

Hypermedia affords communication



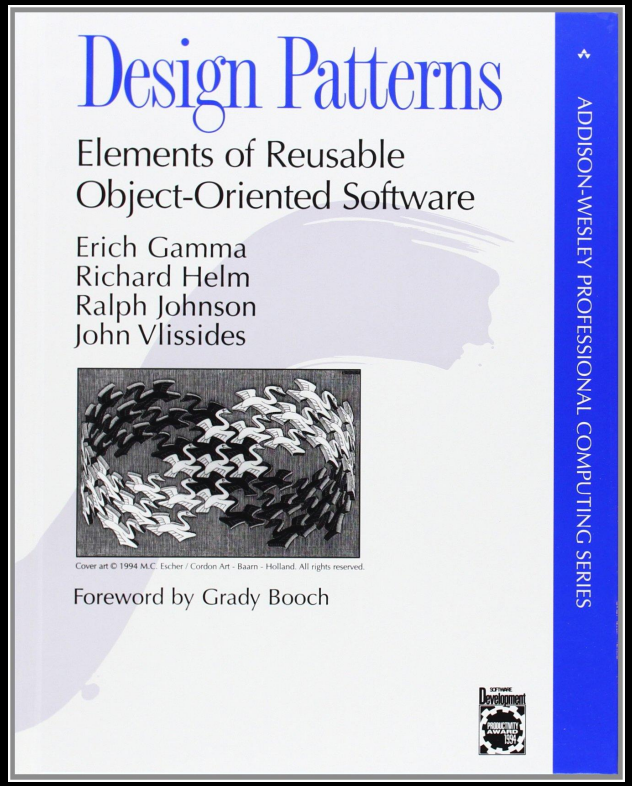
Patterns for Hypermedia

Architectural Patterns



Christopher Alexander, 1977

Patterns are typically applied to code



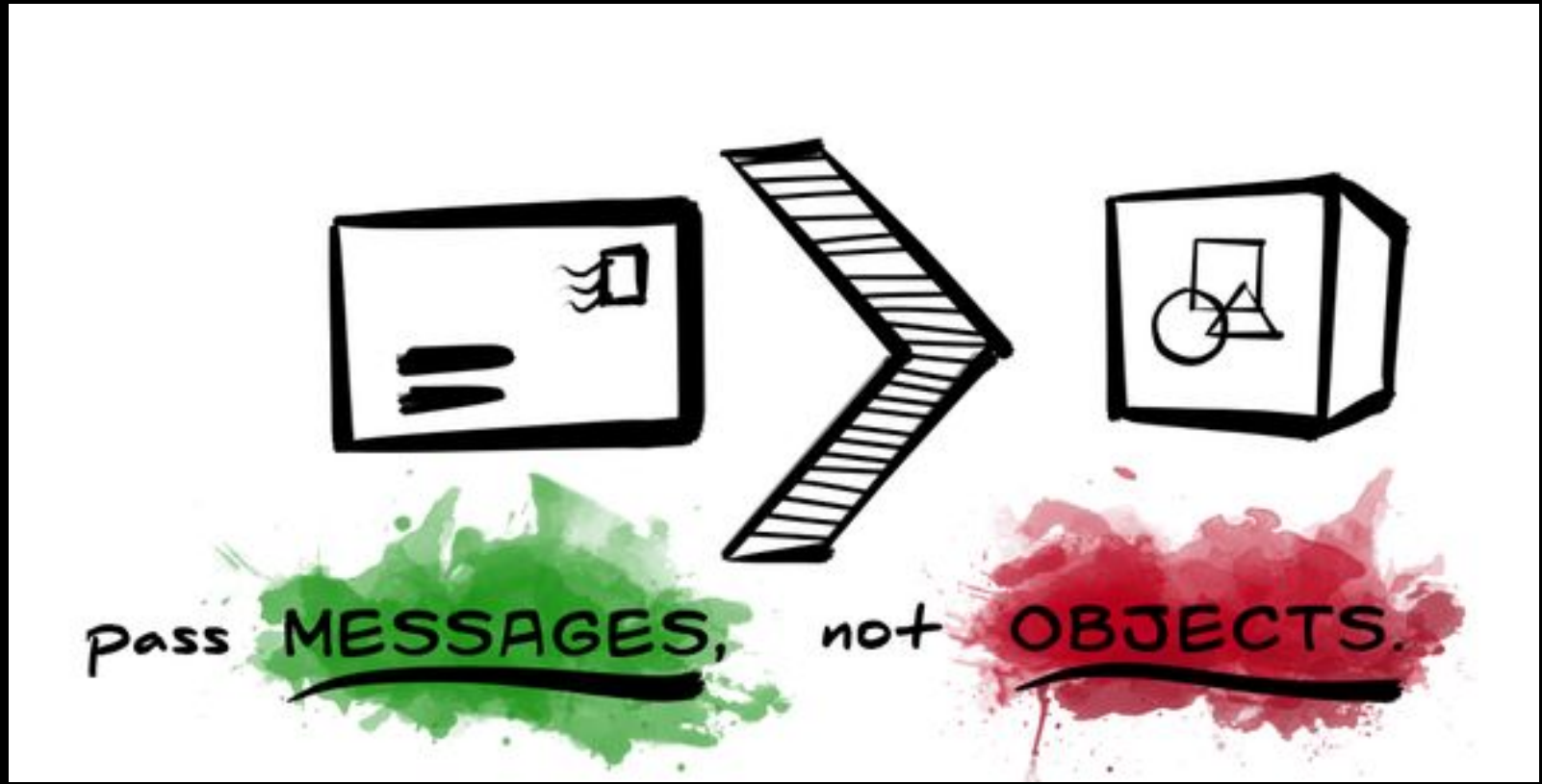
Gang of Four, 1994

Applying patterns to messages

Twelve Patterns for Adaptable Apps

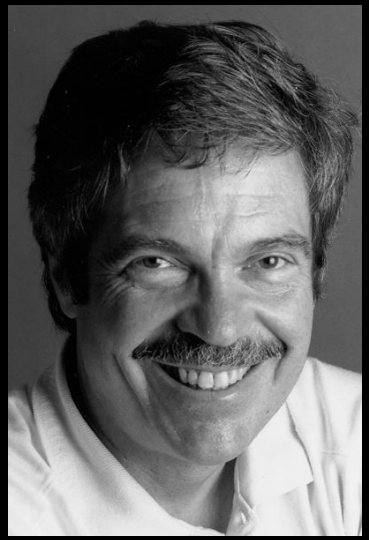
Four Design Patterns
Four Basic Principles
Four Shared Agreements

Design Patterns


pass MESSAGES, not OBJECTS.

Pass Messages, Not Objects

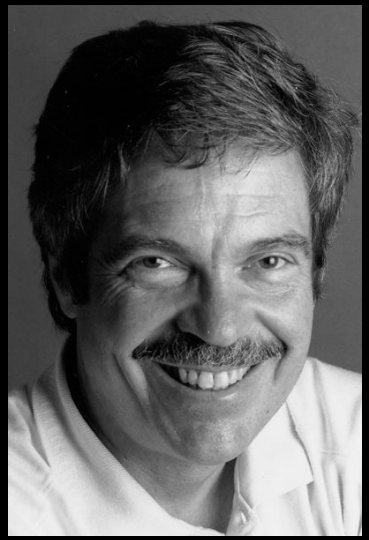
"I'm sorry that coined the term 'objects' for this topic. The big idea is 'messaging'."



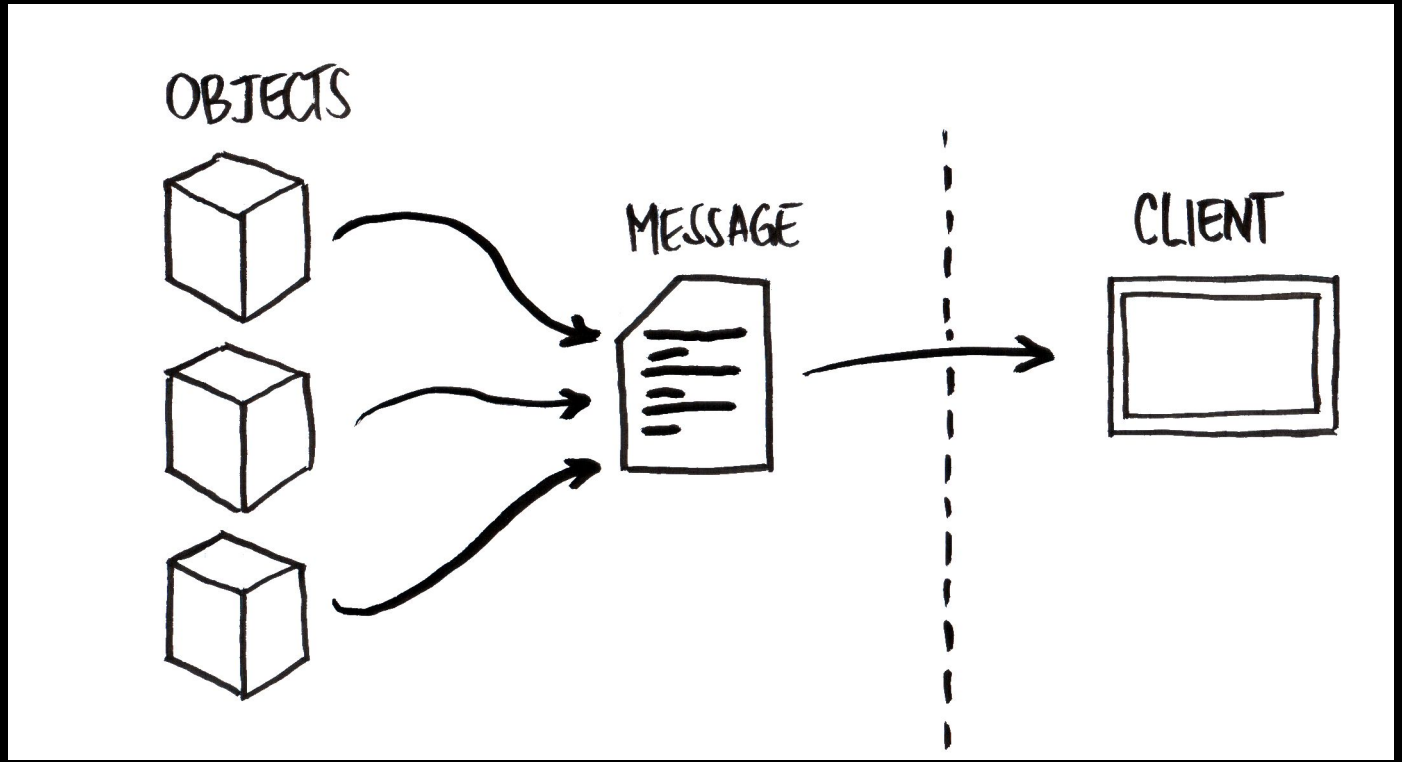
Alan Kay, 1998

Pass Messages, Not Objects

*"I'm sorry that coined the term 'objects' for this topic. **The big idea is 'messaging'.**"*



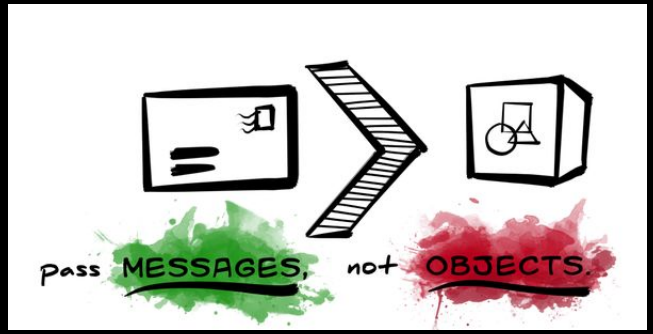
Alan Kay, 1998


OBJECTS
MESSAGE
CLIENT

Pass Messages, Not Objects

Use a Registered Hypermedia Type

HAL
Collection+JSON
Siren
UBER
Atom

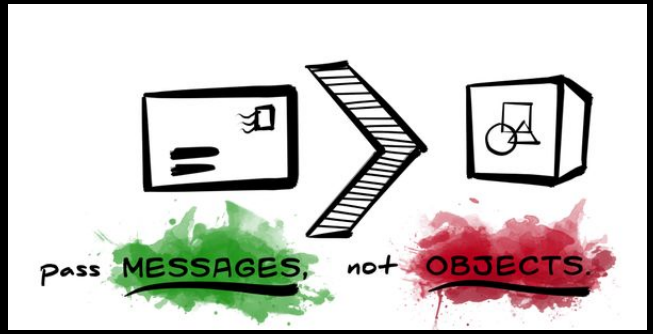


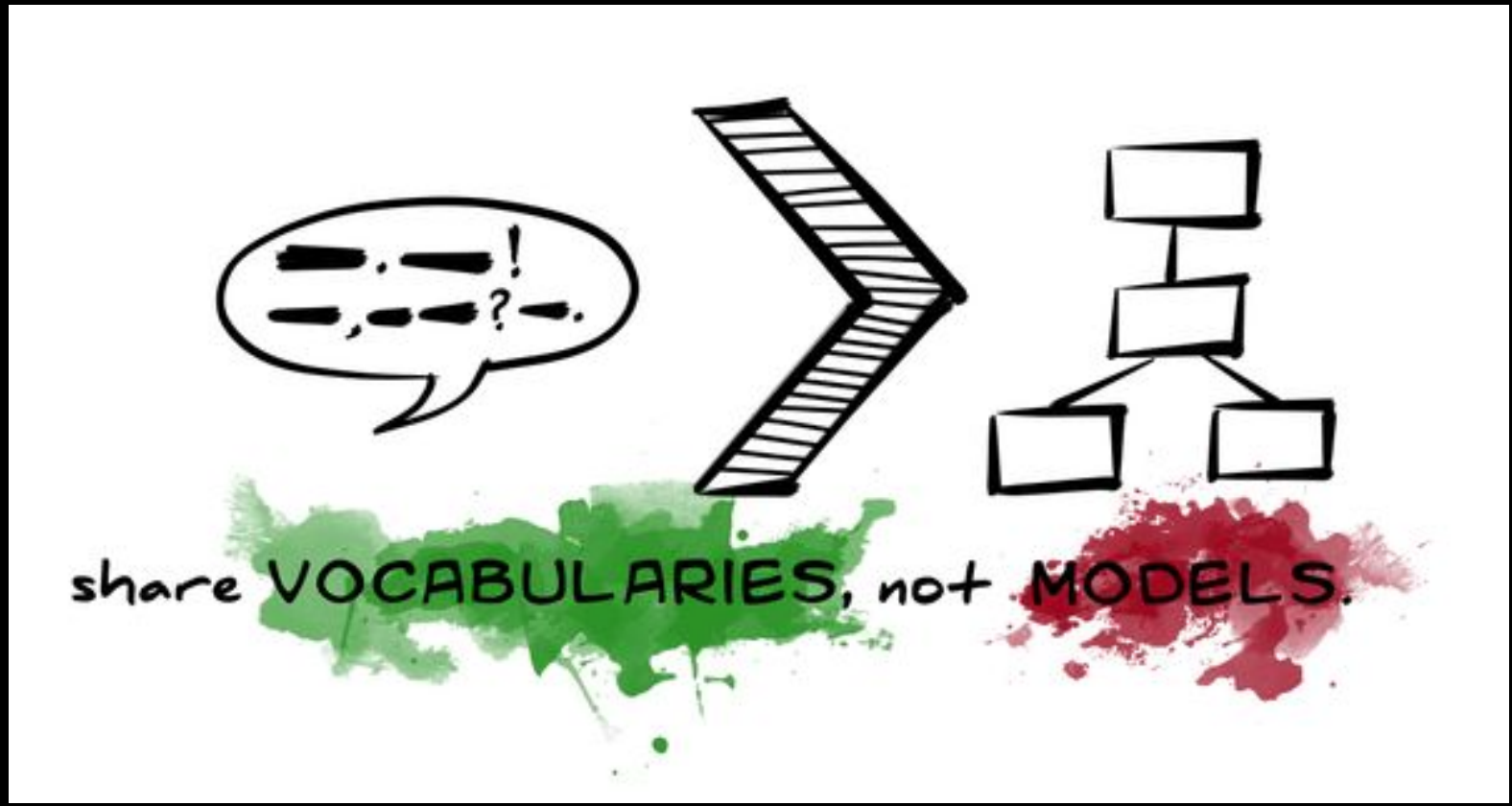
Pass Messages, not Objects

What problem does this solve?

I don’t need to share your object model to interact with you.

Machines can now manage their own internal models independently.




share VOCABULARIES, not MODELS.

Share Vocabularies, Not Models

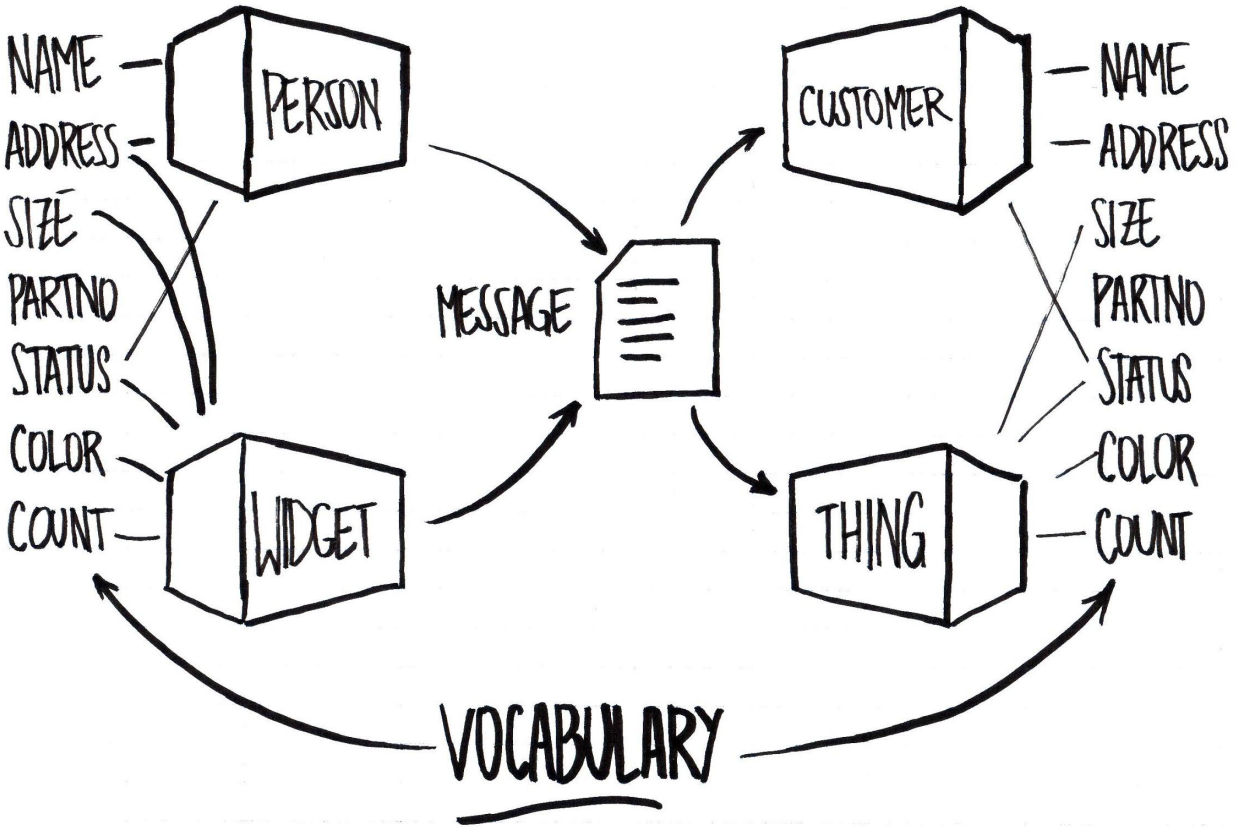
"It is easier to standardize representation and relation types than objects and object-specific interfaces."

-- Roy Fielding

Share Vocabularies, Not Models

*"It is easier to standardize **representation** and **relation** types than objects and object-specific interfaces."*

-- Roy Fielding

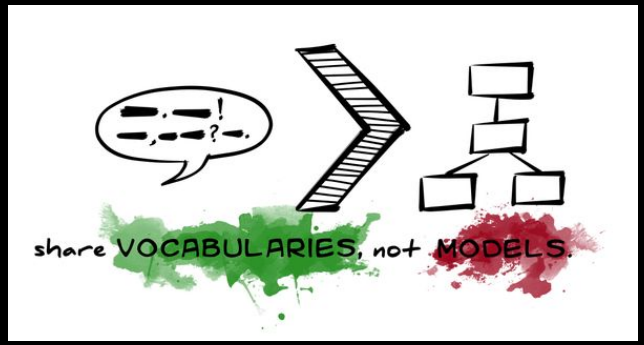


SHARE VOCABULARIES

Share Vocabularies, Not Models

Use Existing Shared Vocabularies

IANA Link Relation Values
Schema.org
Microformats
Dublin Core
Activity Streams

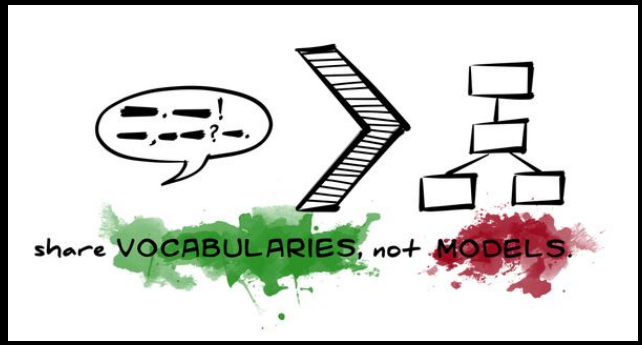


Share Vocabularies, Not Models

What problem does this solve?

Vocabulary is how we “evaluate and select”

Machines can now
evaluate and select without
direct human interaction.



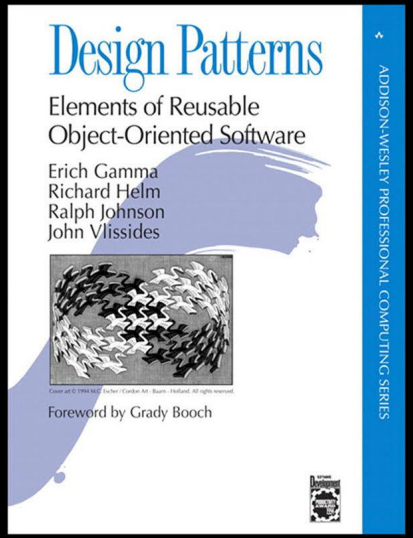
on behalf of...

use REPRESENTER

Use the Representor Pattern

"The Strategy Pattern lets the algorithm vary independently of the clients that use it."

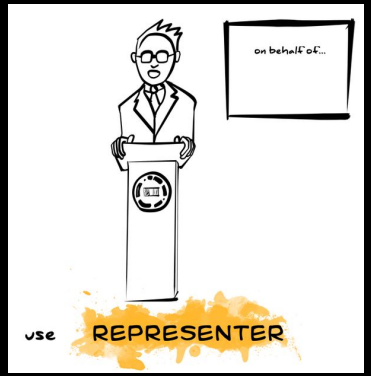
- Gamma, et al.

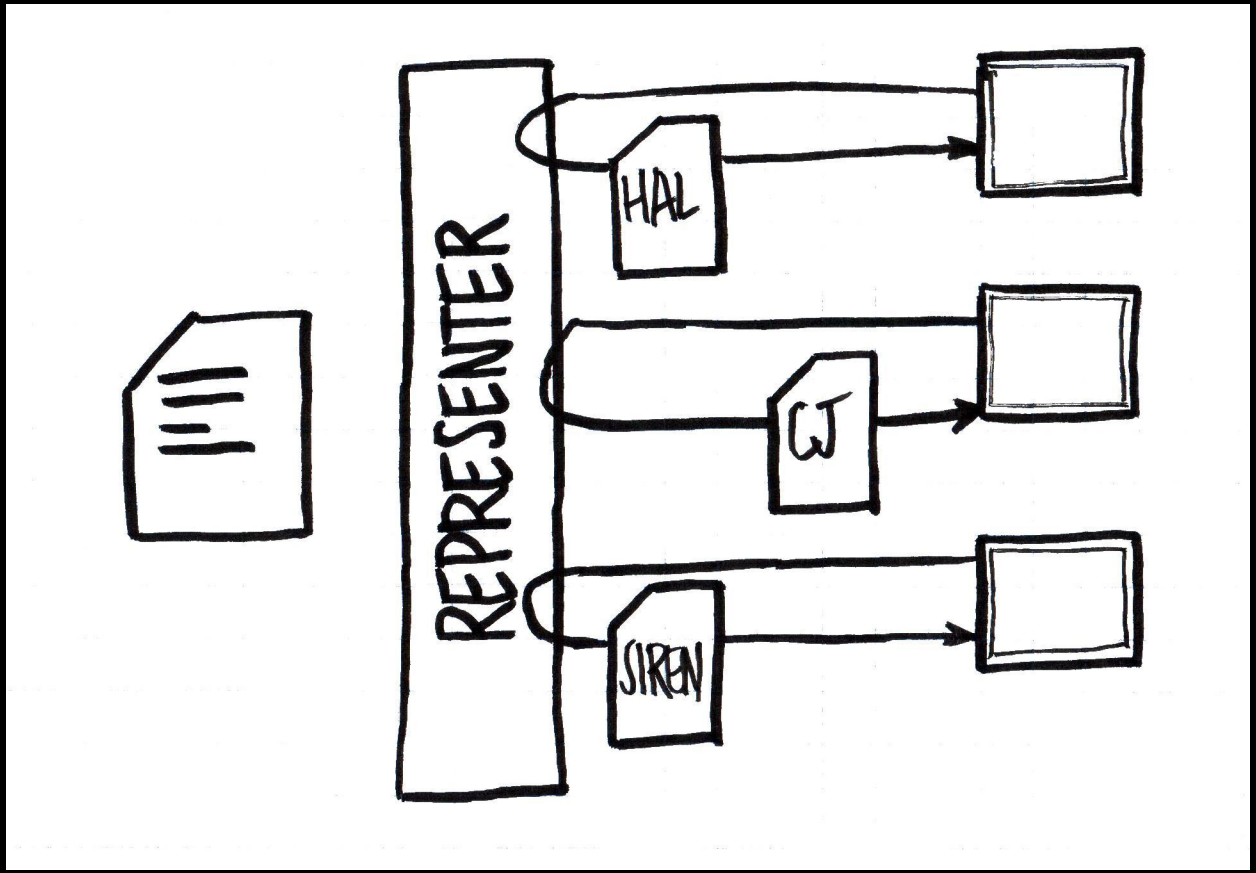


Use the Representor Pattern

Implement a Representor/Strategy Pattern

Standard Internal Resource Model
Strategy Messages Format Dispatch

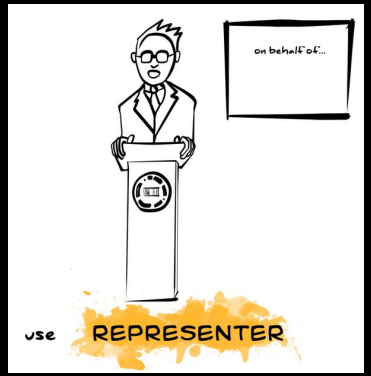



REPRESENTER
HAL
CJ
SIREN

Use the Representor Pattern

Implement a Representor/Strategy Pattern

Standard Internal Resource Model
Strategy Messages Format Dispatch

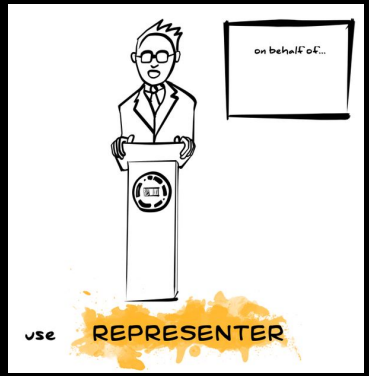


Use the Re n

Implement a Pattern

Standard Inte
Strategy Mes ch

```
// dispatch to requested representor
switch(mimeType.toLowerCase()) {
  case "application/json":
    doc = json(object, root);
    break;
  case "application/vnd.collection+json":
    doc = cj(object, root);
    break;
  case "application/hal+json":
    doc = haljson(object, root);
    break;
  case "application/vnd.uber+xml":
    doc = uberxml(object, root);
    break;
  case "text/html":
  case "application/html":
  default:
    doc = html(object, root);
    break;
}

return doc;
```

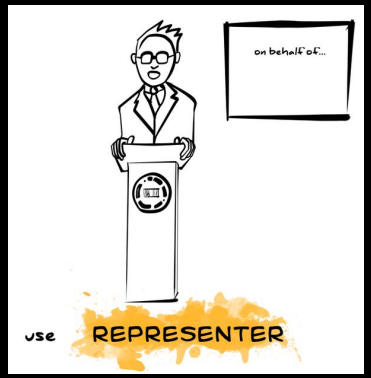


Use the Representor Pattern

What problem does this solve?

Sometimes we need to translate our conversations in order to communicate.

Machines can now “negotiate” the language of a conversation.



publish PROFILES

Publish Profiles

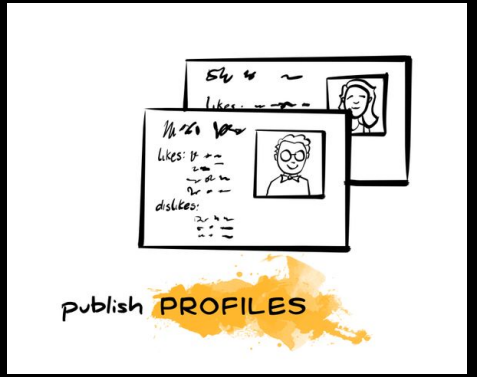
"Profiles provide a way to create a ubiquitous language for talking about APIs (resources) for both humans and machines."

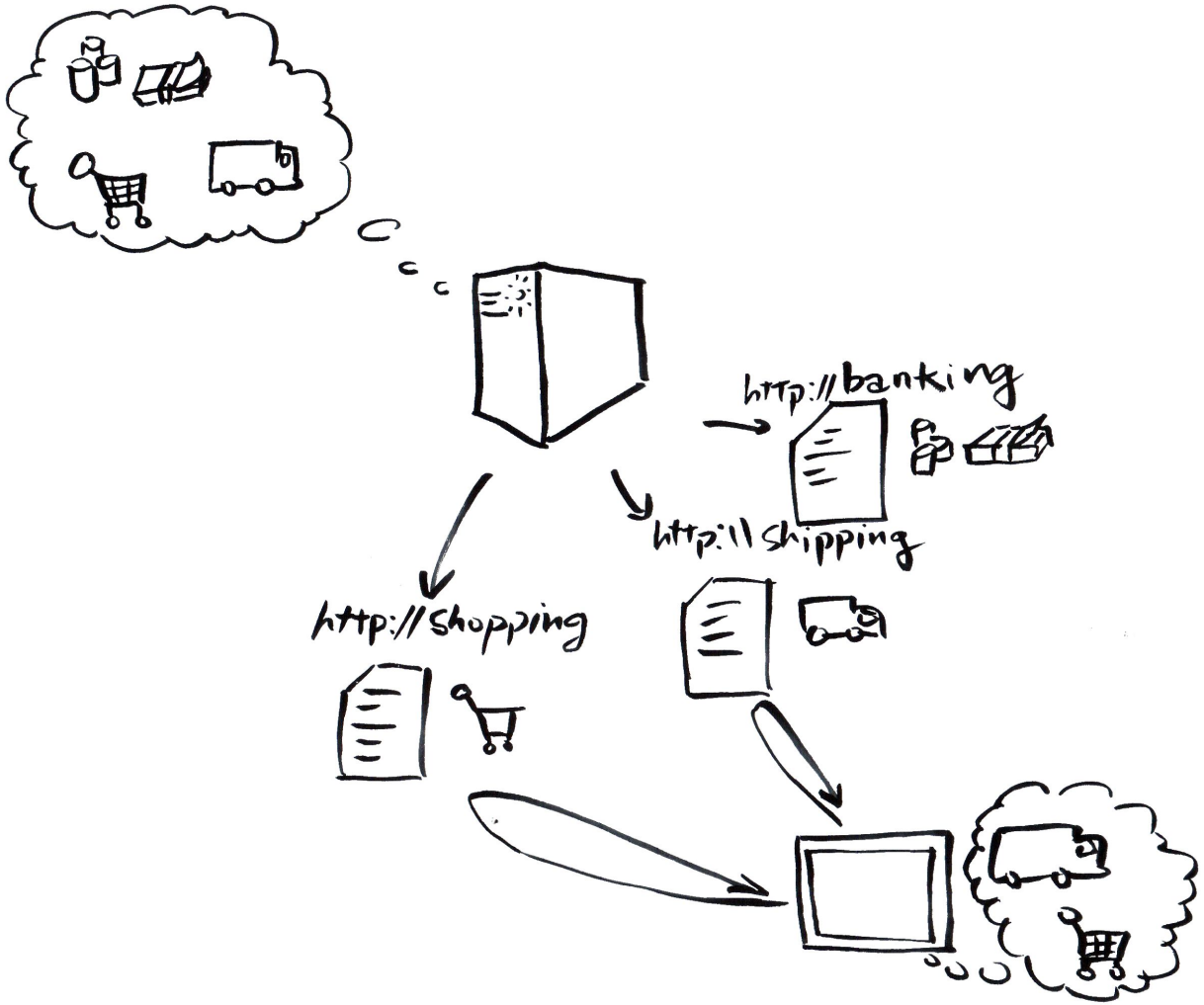
-- Mark Foster

Publish Profiles

Use a Profile like ALPS to share vocabularies

Define all possible data and actions
Publish using Profile Standard (RFC6906)
Servers emit profile URI
Clients validate profile URI

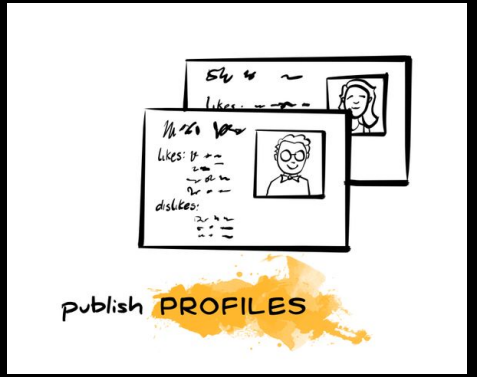



http://banking
http://shipping
http://shopping

Publish Profiles

Use a Profile like ALPS to share vocabularies

Define all possible data and actions
Publish using Profile Standard (RFC6906)
Servers emit profile URI
Clients validate profile URI



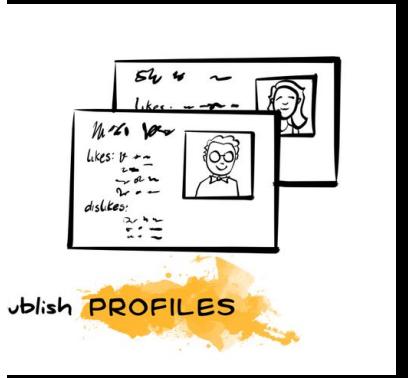
Publish

Use a Pr ularies

Define al
Publish u 06)
Servers
Clients v

products-alps.xml

```
<alps version="1.0">
  <link rel="help" href="http://example.org/documentation/products.html" />
  <doc>
    This is a prototype product API.
  </doc>

  <!-- transitions -->
  <descriptor id="item" type="safe" rt="#product">
    <doc>Retrieve A Single Product</doc>
  </descriptor>

  <descriptor id="collection" type="safe" rt="#product">
    <doc>Provides access to all products</doc>
  </descriptor>

  <descriptor id="search" type="safe" rt="#product">
    <doc>Provides access to all products</doc>
    <descriptor href="#id" />
  </descriptor>

  <descriptor id="edit" type="idempotent" rt="#product">
    <doc>Updates A Product</doc>
    <descriptor href="#product" />
  </descriptor>

  <descriptor id="create" type="unsafe" rt="#product">
    <doc>Allows the creation of a new product</doc>
    <descriptor href="#product" />
  </descriptor>
```

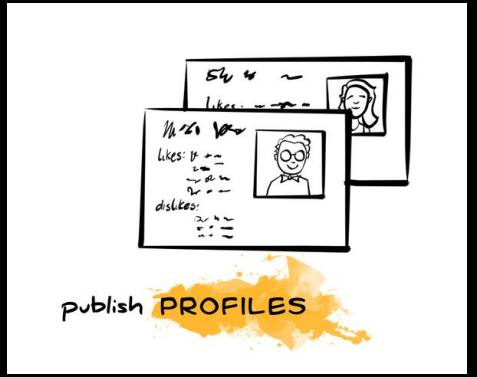


Publish Profiles

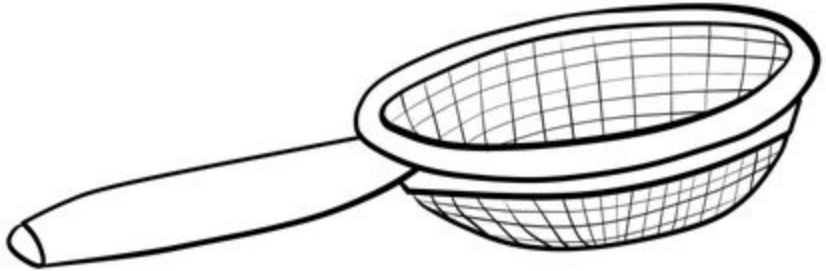
What problem does this solve?

I need to know what we're talking about.

Machines can now
validate domain topics easily



Basic Principles

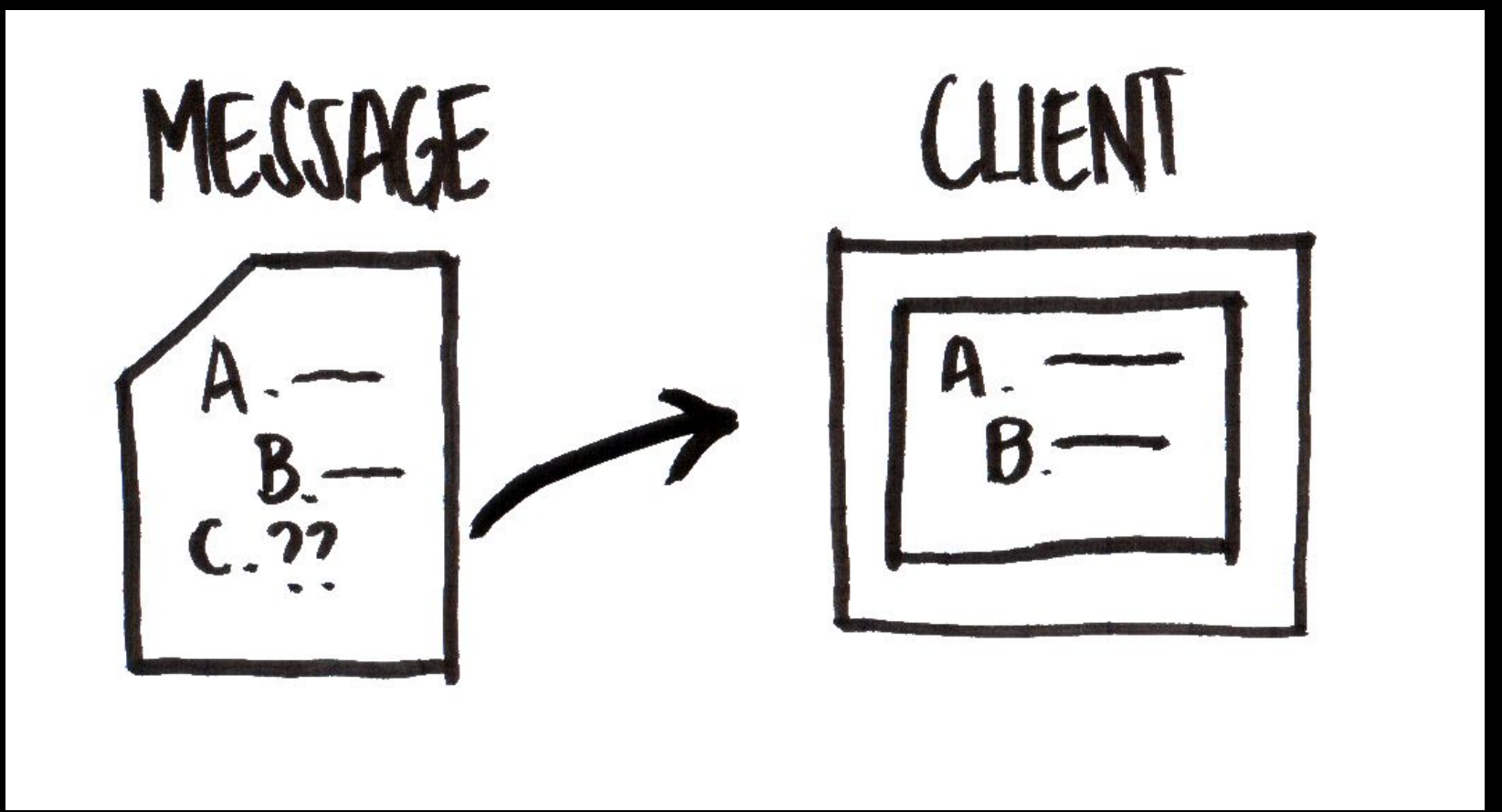


must IGNORE

Must Ignore

“The main goal of the MUST IGNORE pattern of extensibility is to allow backwards- and forwards-compatible changes.”

- David Orchard


MESSAGE
A
B
C ??
CLIENT
A
B

Must Ignore

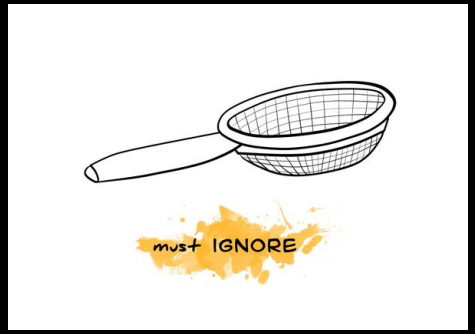
Clients MUST IGNORE any data/inputs that the client does not understand.

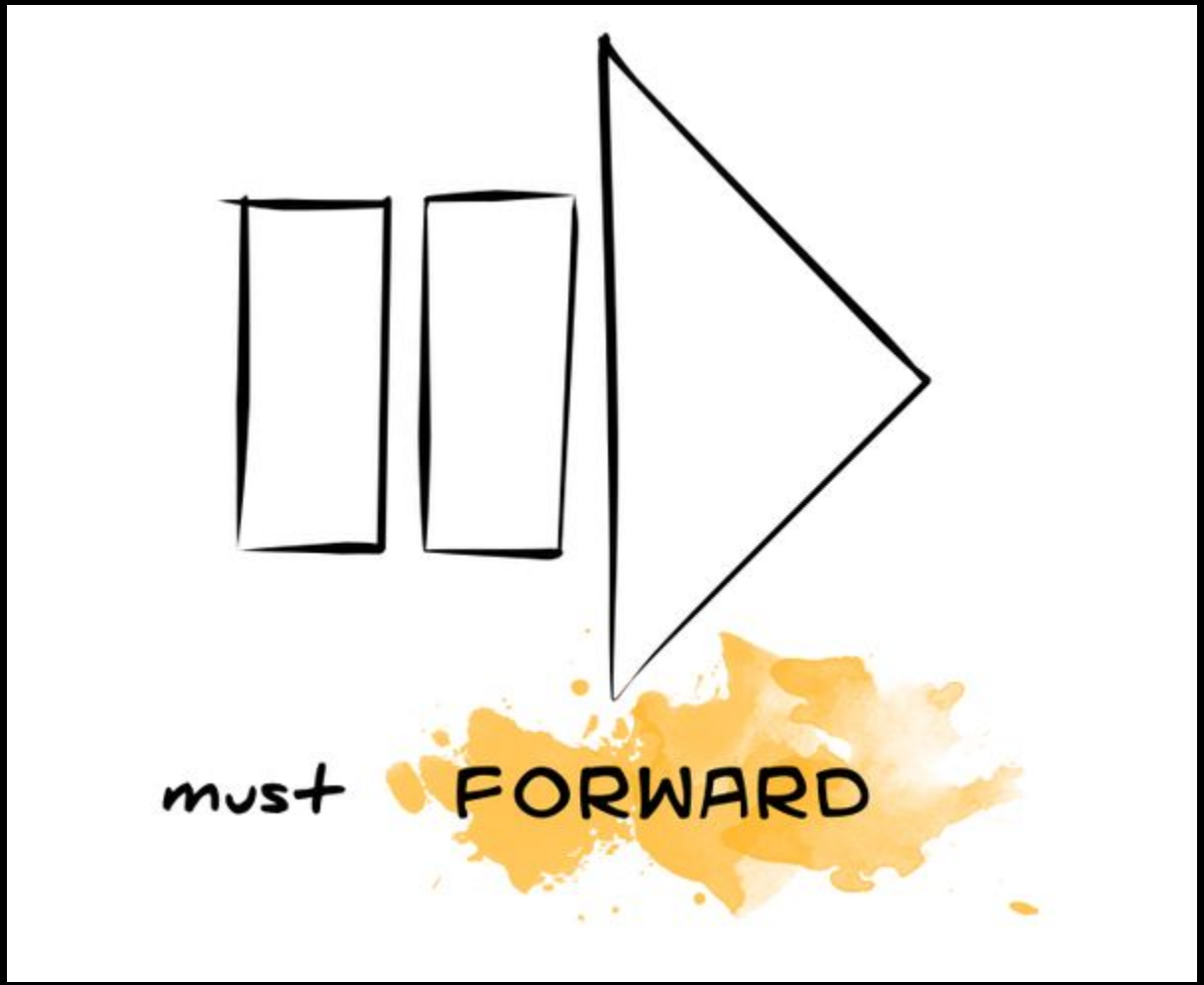
Must Ignore

What problem does this solve?

Ignoring what we don't understand lets us "do our own thing" w/o knowing everyone's job

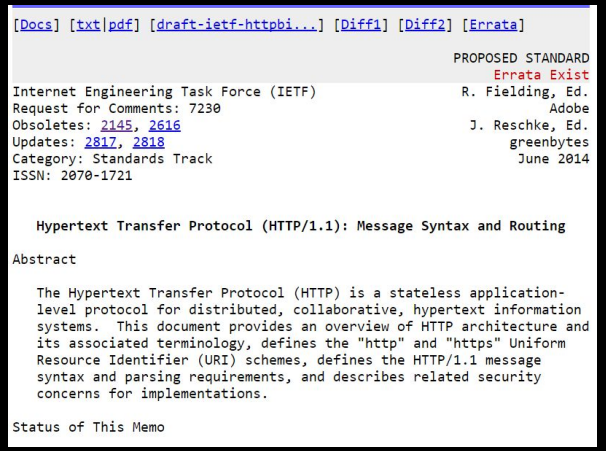
Machines can now focus on their own job, not everyone's job.




must
FORWARD

MUST FORWARD

“A proxy MUST forward unrecognized header fields…”
-- RFC 7230



[Docs] [txt|pdf] [draft-ietf-httpbi...] [Diff1] [Diff2] [Errata]

PROPOSED STANDARD
Errata Exist

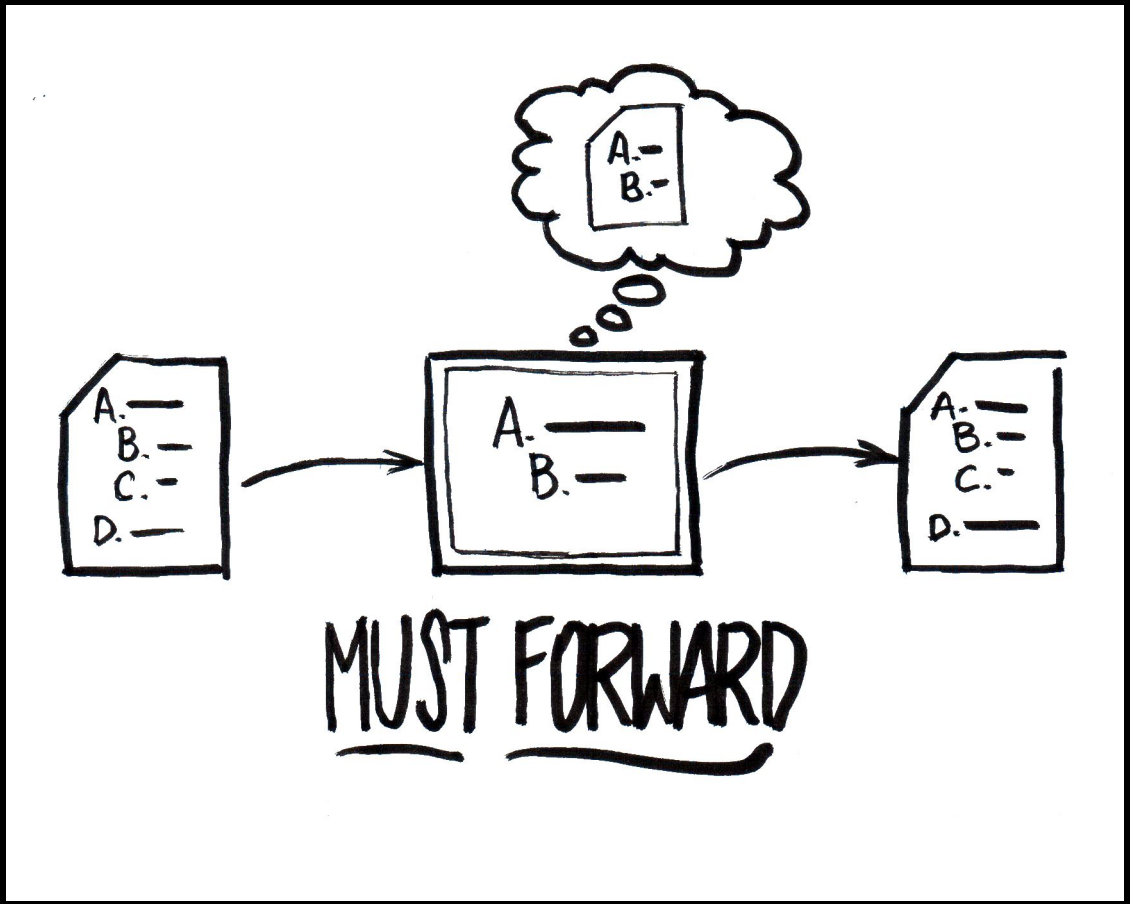
Internet Engineering Task Force (IETF) — R. Fielding, Ed., Adobe
Request for Comments: 7230 — J. Reschke, Ed., greenbytes
Obsoletes: 2145, 2616
Updates: 2817, 2818
Category: Standards Track — June 2014
ISSN: 2070-1721

Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing

Abstract

The Hypertext Transfer Protocol (HTTP) is a stateless application-level protocol for distributed, collaborative, hypertext information systems. This document provides an overview of HTTP architecture and its associated terminology, defines the "http" and "https" Uniform Resource Identifier (URI) schemes, defines the HTTP/1.1 message syntax and parsing requirements, and describes related security concerns for implementations.

Status of This Memo


A.
B.
A.
B.
A.
B.
C.
D.
A.
B.
C.
D.
MUST FORWARD

Must Forward

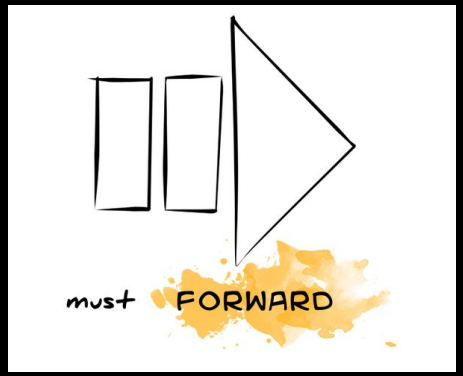
Clients MUST FORWARD (unchanged) any input fields (URL or FORM) that the client does not recognize.

Must Forward

What problem does this solve?

We don’t edit for others around us.

Machines can now co-operate w/o full understanding of other’s work



provide MRU

Provide MRU

“A feature of convenience allowing users to quickly see and access the last few used files and documents.”
-- Wikipedia

Common menus in Microsoft Windows

From Wikipedia, the free encyclopedia

This is a **list of commonly used Microsoft Windows menus**.

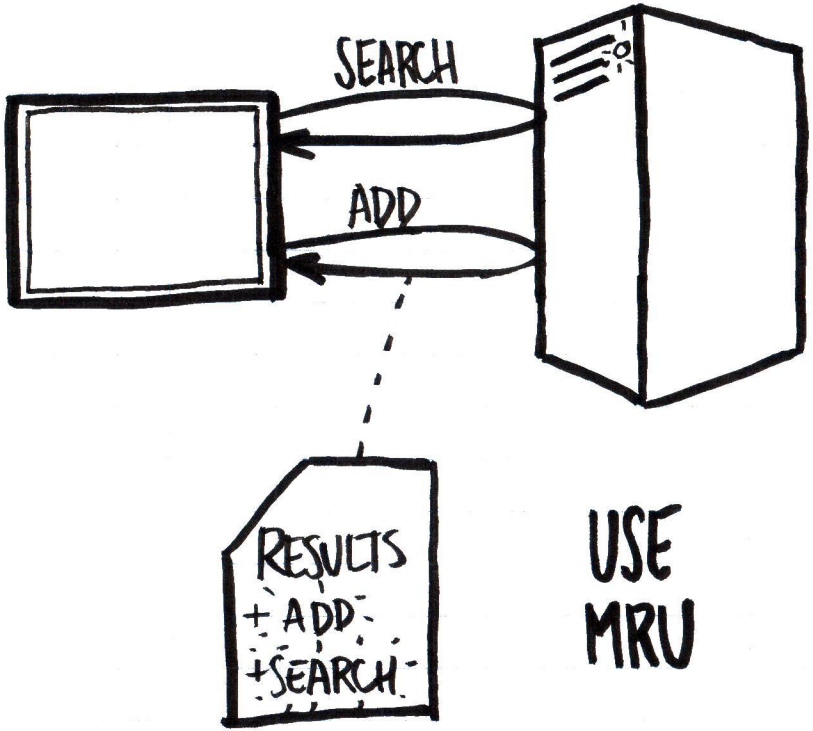
Contents [hide]

1 Microsoft menus
1.1 Most Recently Used menu
1.2 Properties menu
1.3 System menu
2 References

Microsoft menus [edit]

Most Recently Used menu [edit]

Most Recently Used (MRU) is a term used in computing to refer to the list of progr
quickly see and access the last few used files and documents, but could also be c


SEARCH
ADD
RESULTS
+ ADD
+ SEARCH
USE
MRU

Provide MRU

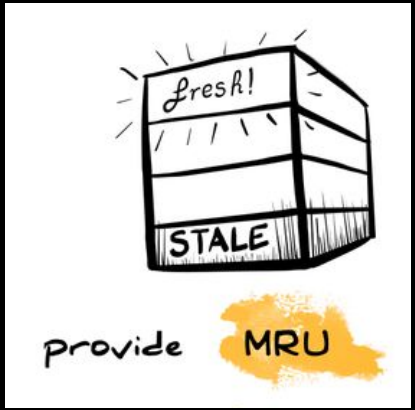
Services SHOULD return the most recently-used (MRU) LINKS and FORMS in all responses.

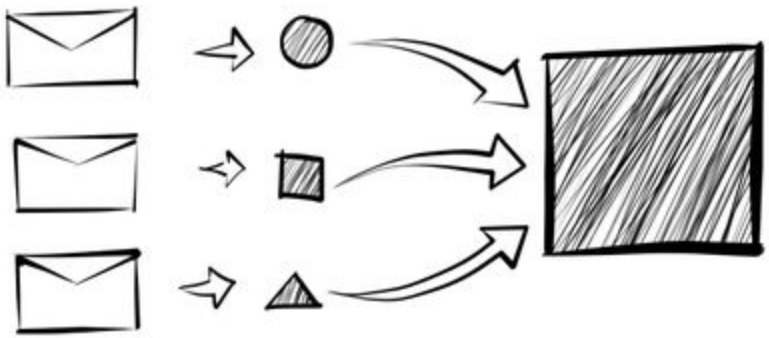
Provide MRU

What problem does this solve?

We need most-used tools close at hand

Machines can now find most-used affordances easily

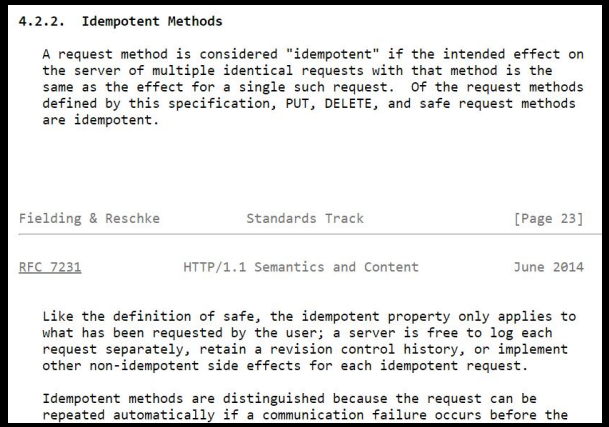




use IDEMPOTENCY

Use Idempotence

“Can be applied multiple times without changing the result beyond the initial application.”
-- Wikpedia

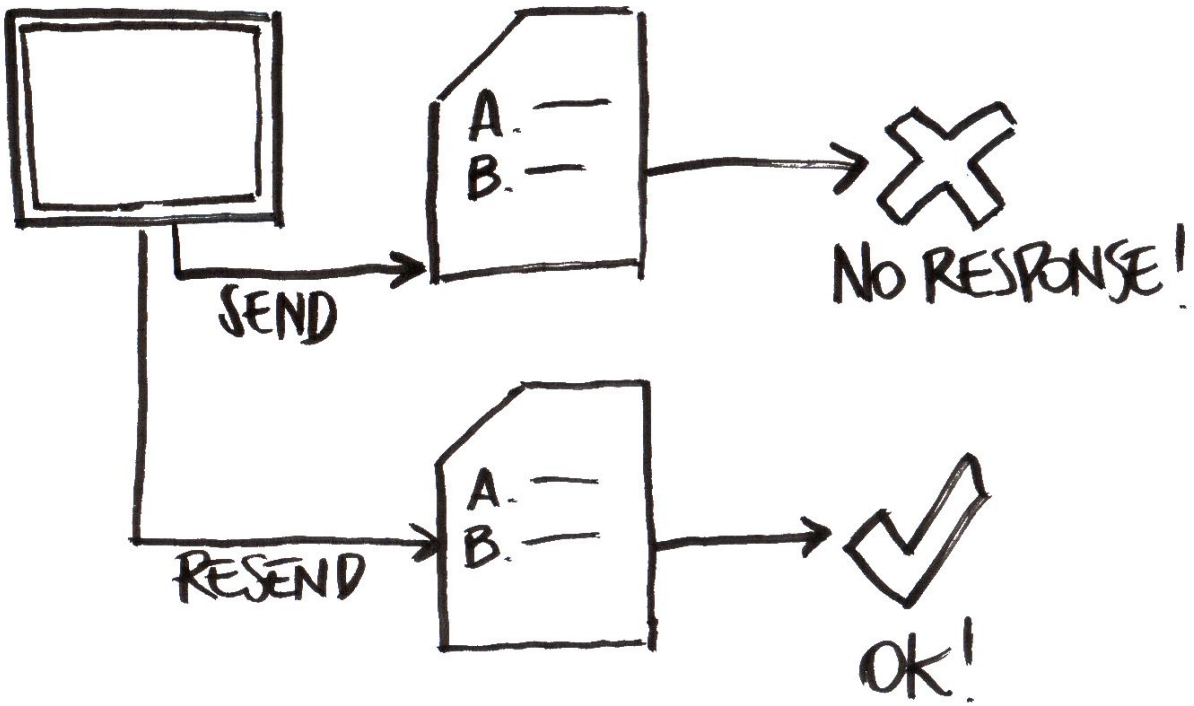


4.2.2. Idempotent Methods

A request method is considered "idempotent" if the intended effect on the server of multiple identical requests with that method is the same as the effect for a single such request. Of the request methods defined by this specification, PUT, DELETE, and safe request methods are idempotent.

Like the definition of safe, the idempotent property only applies to what has been requested by the user; a server is free to log each request separately, retain a revision control history, or implement other non-idempotent side effects for each idempotent request.

Idempotent methods are distinguished because the request can be repeated automatically if a communication failure occurs before the


A.
B.
SEND
NO RESPONSE!
A.
B.
RESEND
OK!

Use Idempotence

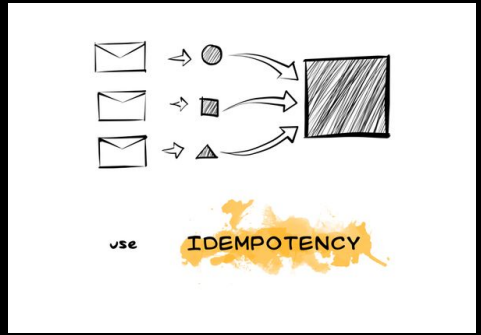
All network requests SHOULD be idempotent in order to allow clients to safely repeat them when response is unclear.

Use Idempotence

What problem does this solve?

If things didn’t work right the first time, we need to try again.

Machines can now safely “try again”

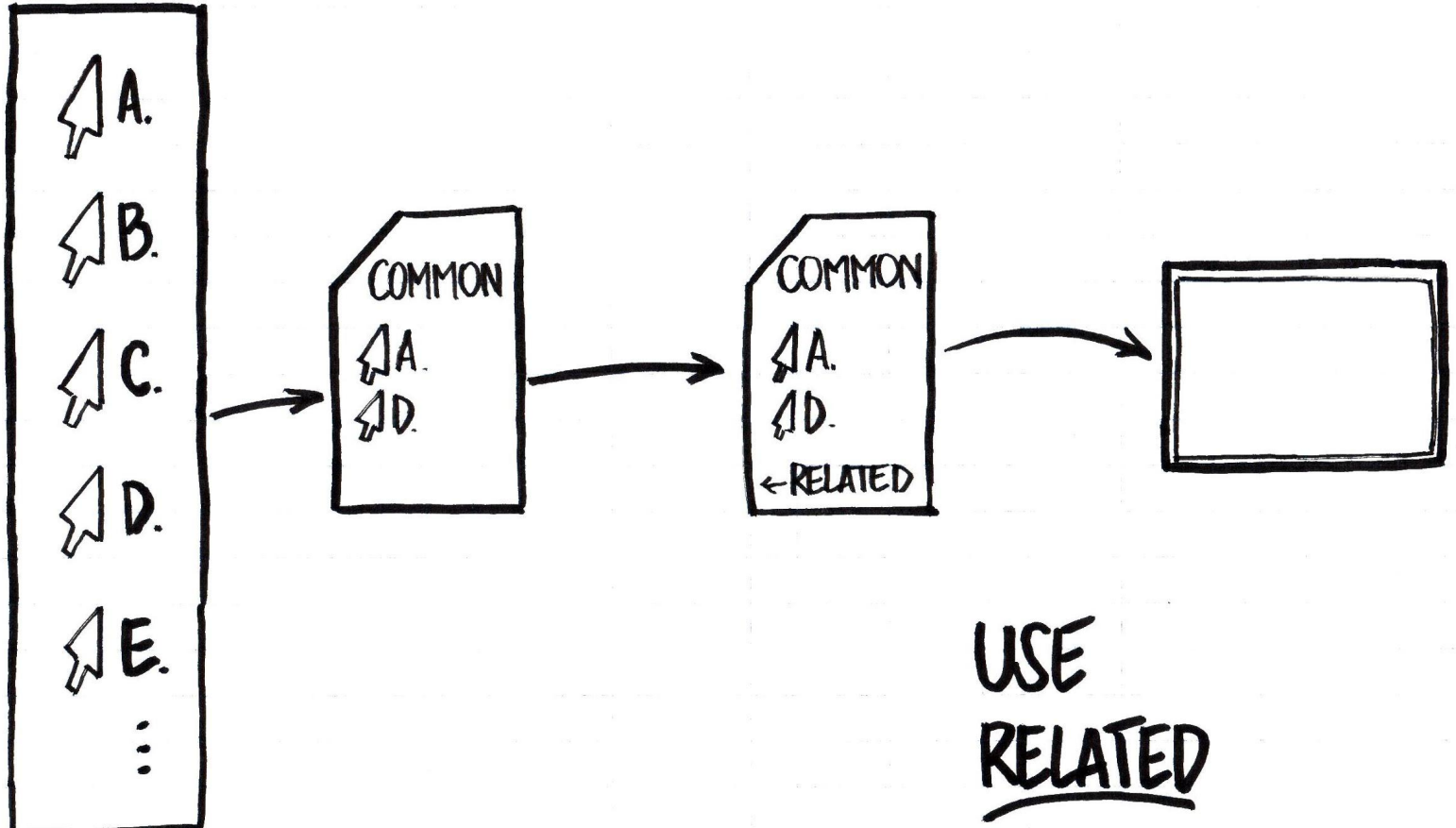


Shared Agreements

use RELATED

Use Related

“By watching what you click on in search results, Google can learn that you favor particular sites.” – Danny Sullivan, 2009


A.
B.
C.
D.
E.
COMMON
A.
D.
COMMON
A.
D.
←RELATED
USE
RELATED

Use Related

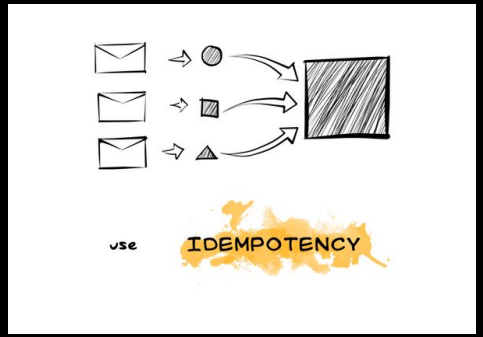
Services SHOULD return a RELATED LINK that responds with ALL the possible actions for this context.

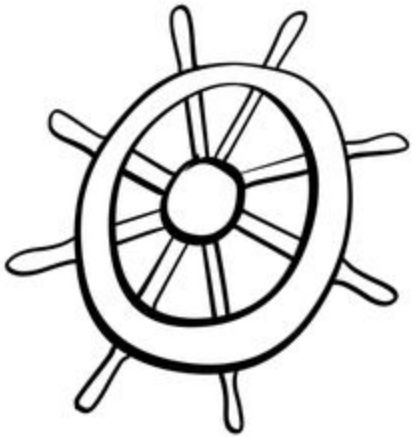
Use Related

What problem does this solve?

I can’t remember everything, need an easy way to look up instructions.

Machines can now “look up” the available affordances.





next

previous

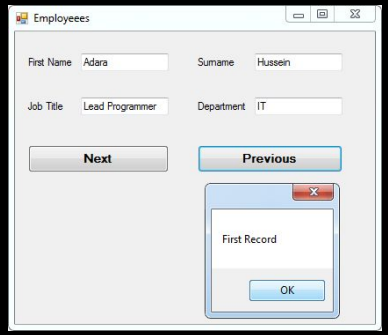
done

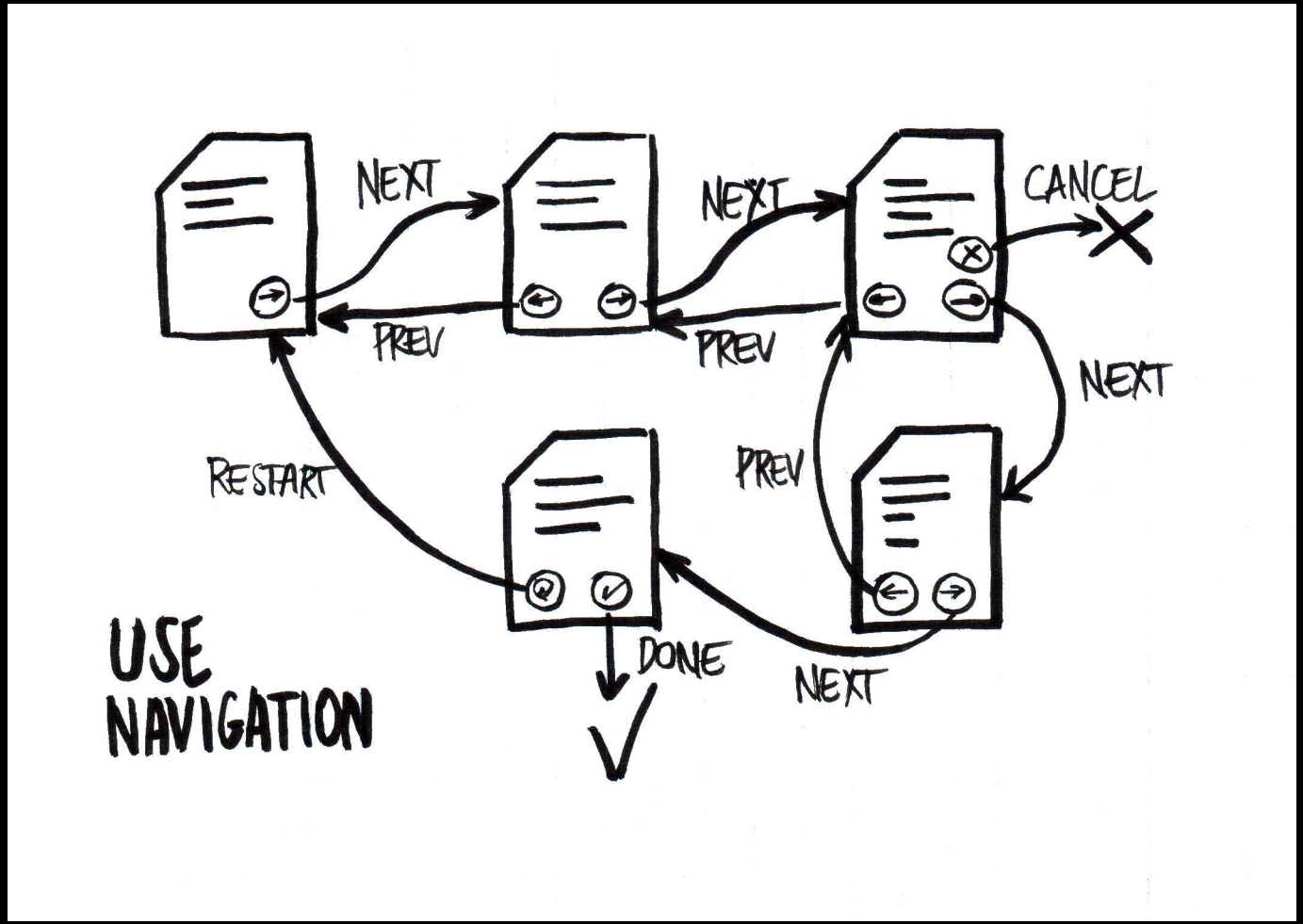
cancel

use NAVIGATION

Use Navigation

“To achieve a single goal which can be broken down into dependable sub-tasks.”
-- Design Patterns (@uipatterns)




NEXT
NEXT
CANCEL
PREV
PREV
NEXT
RESTART
PREV
DONE
NEXT
USE
NAVIGATION

Use Navigation

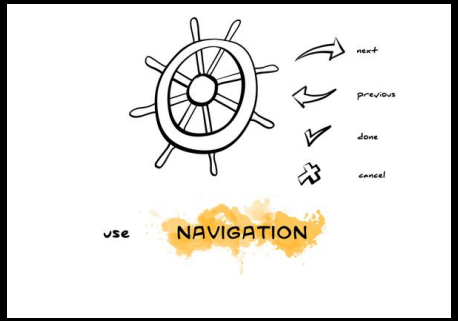
Services SHOULD provide "next/previous" LINK to handle multi-step workflow with "cancel", "restart", & "done."

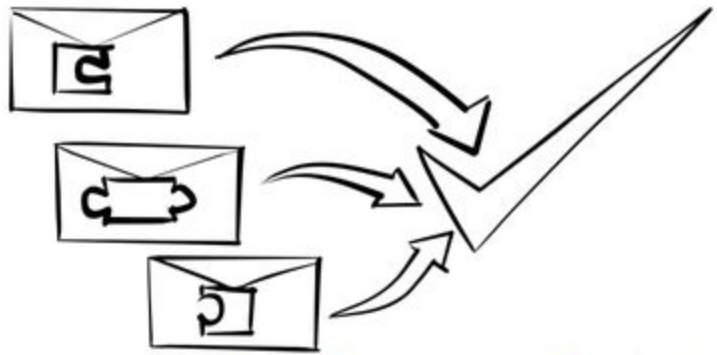
Use Navigation

What problem does this solve?

I can’t keep all the steps in my head

Machines can now navigate through a long series of steps safely.



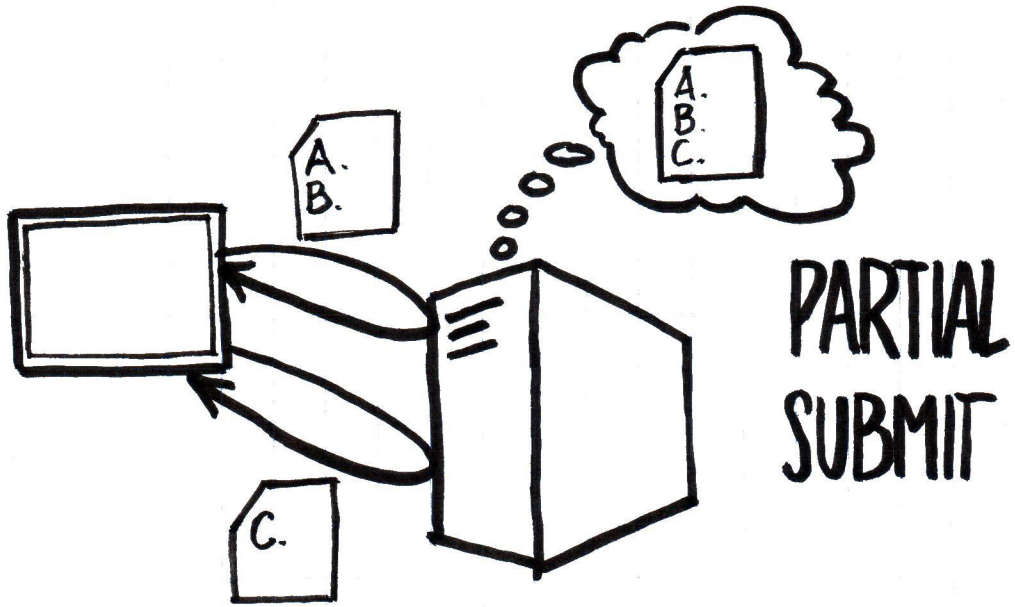


PARTIAL SUBMIT

Partial Submit

“Think of the actions as approximations of what is desired.”

-- Donald Norman



PARTIAL SUBMIT

Partial Submit

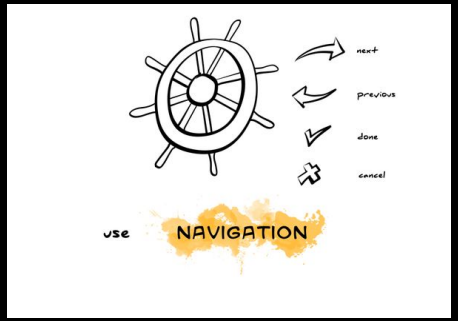
Services SHOULD accept partially filled-in FORM and return a new FORM with the remaining fields.

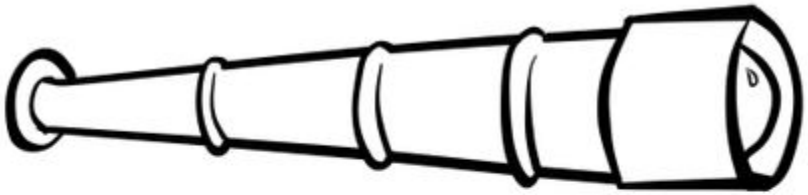
Use Navigation

What problem does this solve?

I sometimes only know part of the story.

Machines can now interact in small parts and not always be perfect.





STATE WATCH

State Watch

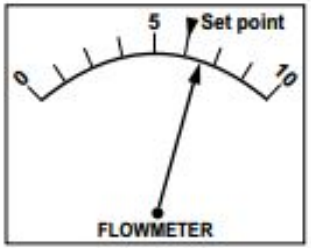
“Data representing variables in a dynamical system…”

-- Jens Rassmussen

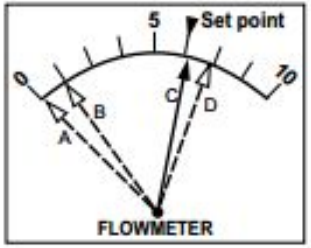
State Wato

“Data rep *a dynamical system…”*

Rassmussen



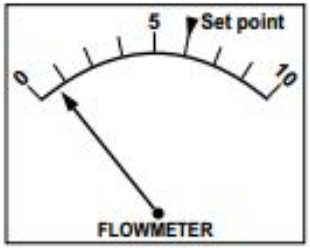
SIGNAL
- Keep at set point
- Use deviation as error signal
- Track continuously



SIGN

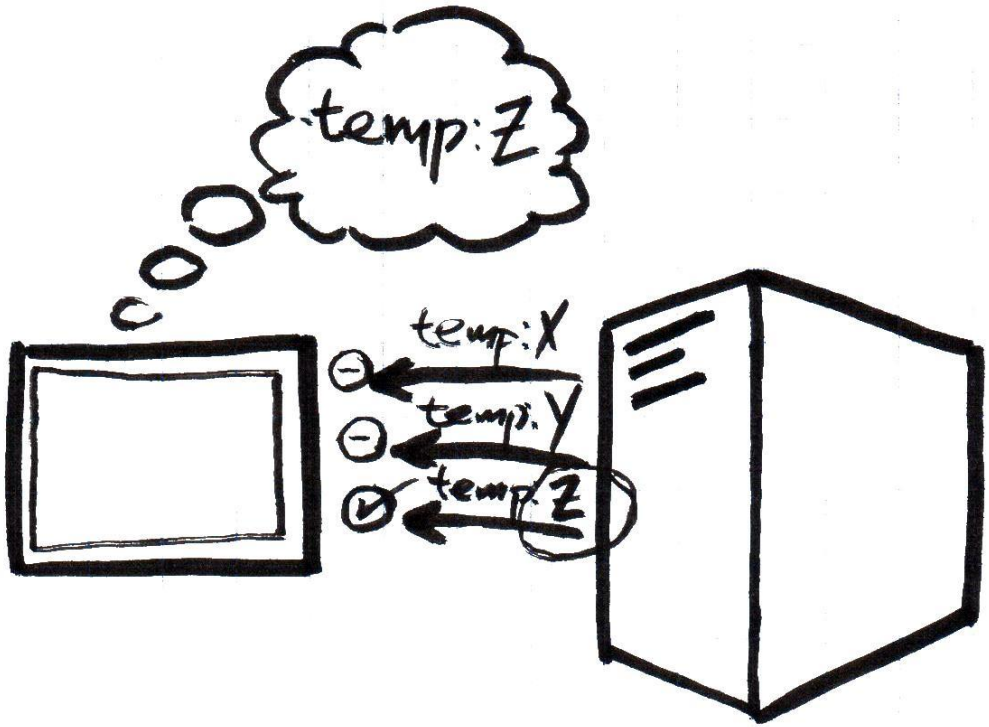
Stereotype acts

If Valve Open	If C, ok If D, adjust flow
If Valve Closed	If A, ok If B, recalibrate meter



SYMBOL

If, after calibration, is still B, begin to read meter and speculate functionally (could be a leak)



STATE WATCH

State Watch

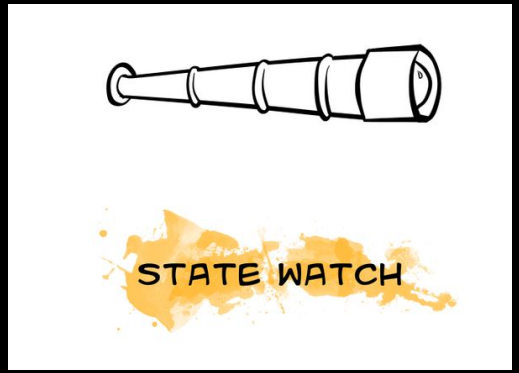
Services SHOULD allow clients to subscribe to WATCH VALUES so that clients can deterimine "done."

Use State Watch

What problem does this solve?

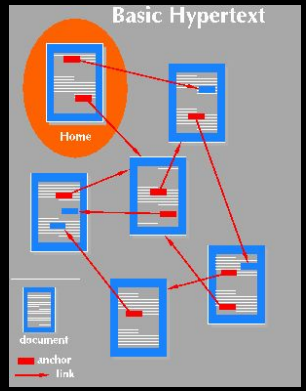
My boss doesn't always set my goals.

Machines can now set their own goals and act accordingly.

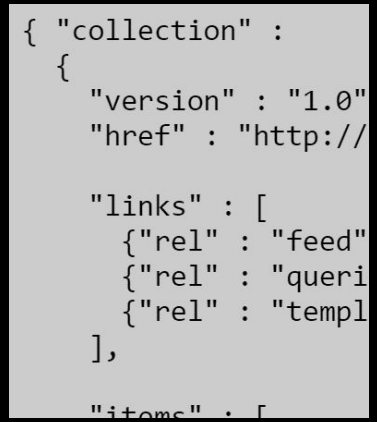


Summary

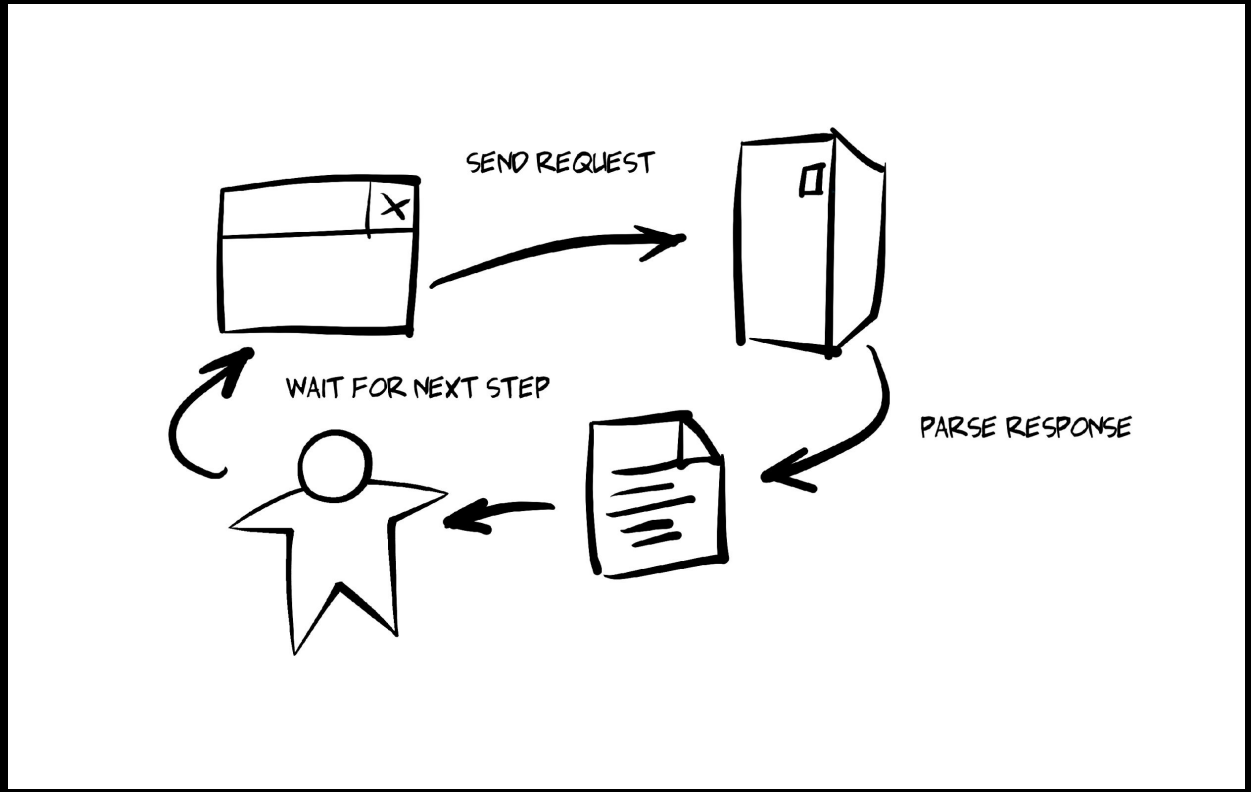
Hypermedia is the language of the WWW



Hypermedia Types are the programming language of the WWW



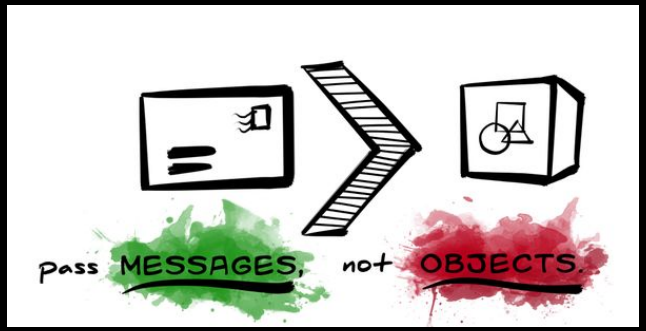
Hypermedia affords communication

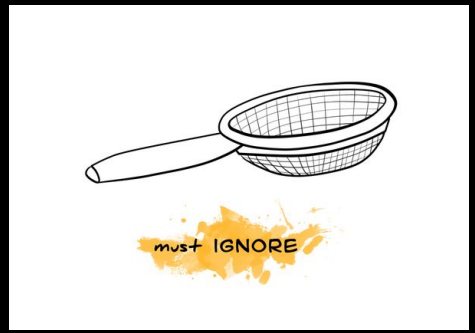


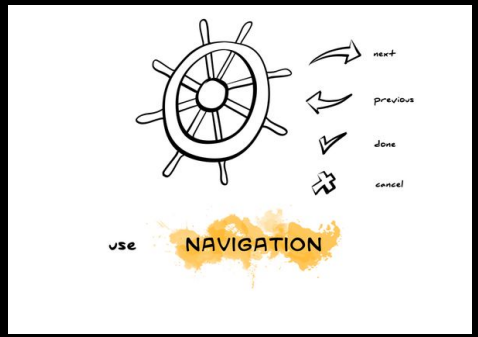
Apply patterns to messages

Twelve Patterns for Adaptable APIs

Four Design Patterns
Four Basic Principles
Four Shared Agreements

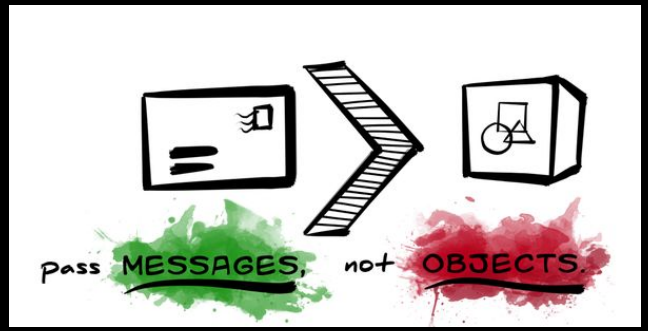






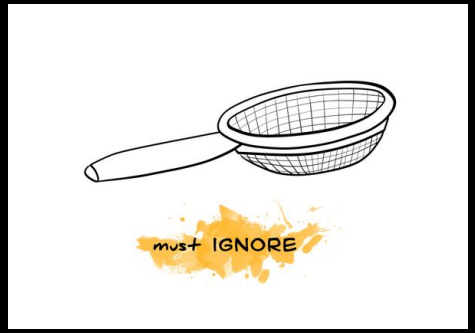
Design Patterns

1. PASS MESSAGES, NOT OBJECTS
2. SHARE VOCABULARIES, NOT MODELS
3. THE REPRESENTOR PATTERN
4. PUBLISH PROFILES



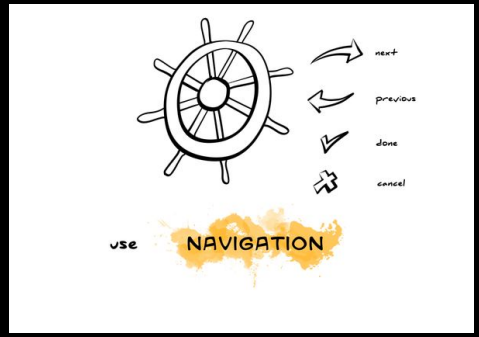
Basic Principles

5. MUST IGNORE
6. MUST FORWARD
7. PROVIDE MRU
8. USE IDEMPOTENCE



Basic Agreements

9. USE RELATED
10. USE NAVIGATION
11. PARTIAL SUBMIT
12. STATE WATCH



The Best Software Architecture

"The best software architecture 'knows' what changes often and makes that easy."

- Paul Clements

Twelve Patterns for Evolvable APIs

Mike Amundsen
API Academy / CA
@mamund

Drawings by Diogo Lucas
@diogoclucas