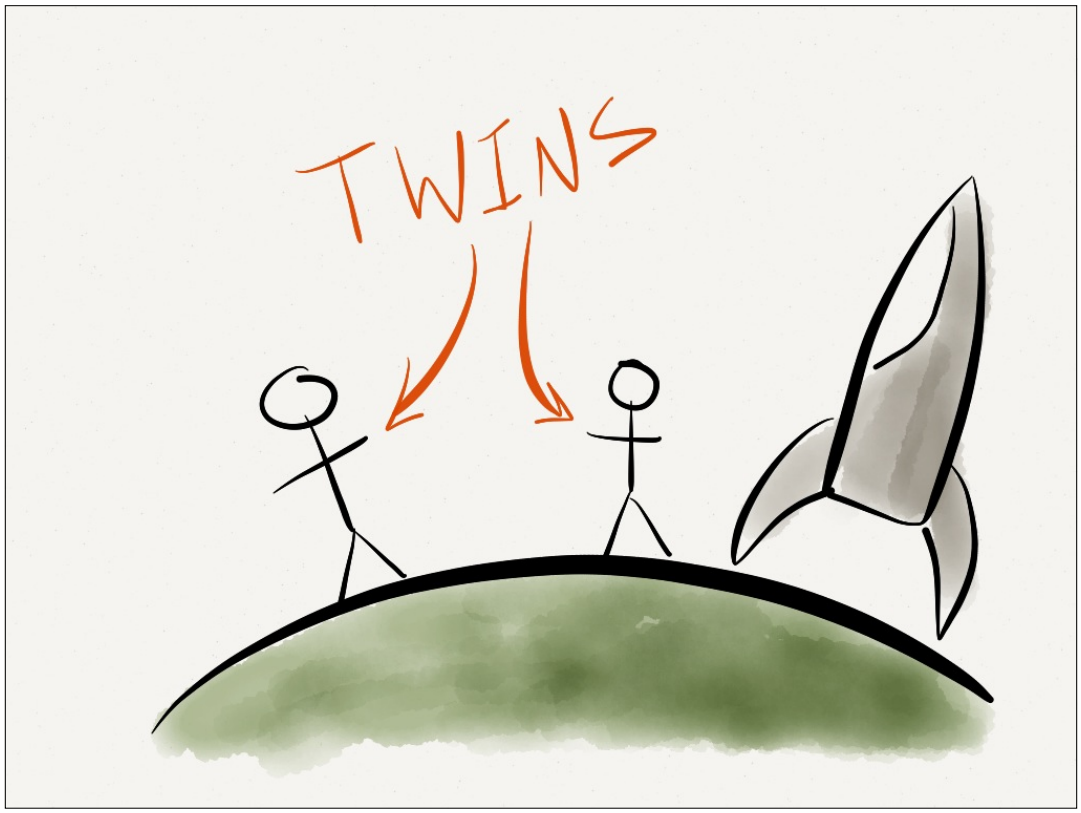
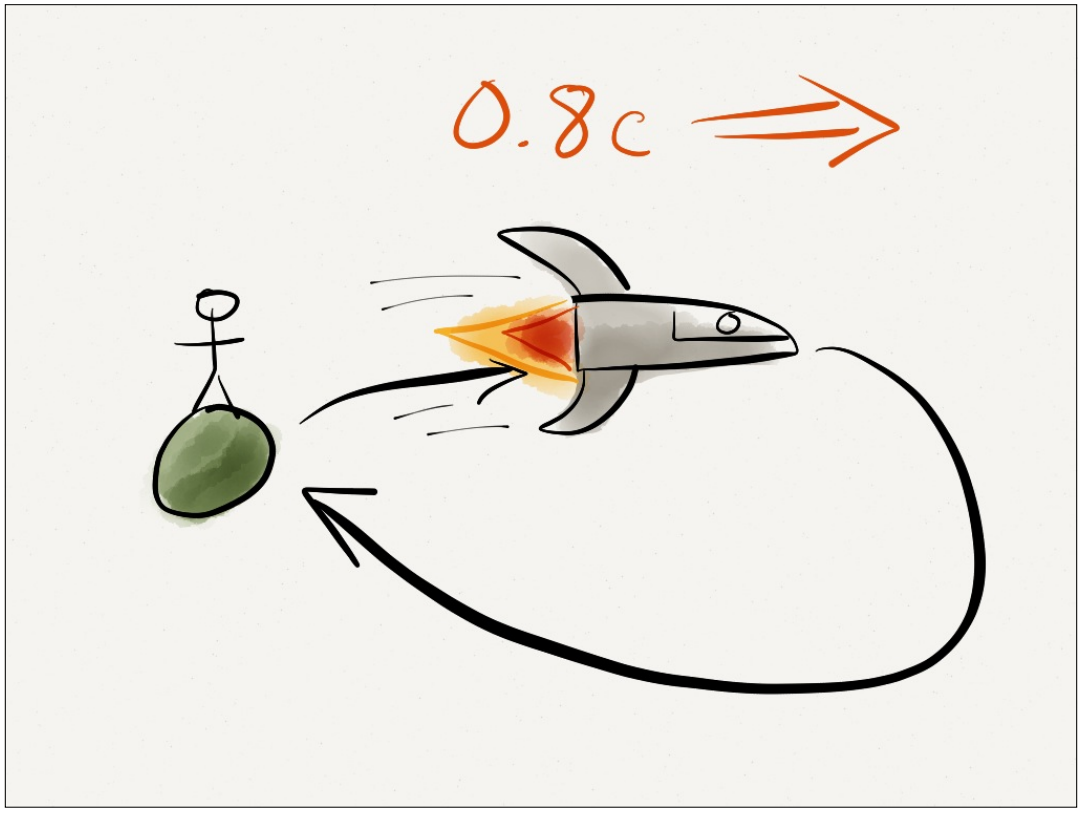
CONNASCENCE EXAMINED

@jimweirich

EdgeCase

TWIN PARADOX


TWINS


0.8c

$$\epsilon = \sqrt{1 - v^2/c^2}$$

$$\sqrt{1 - 0.8^2/1^2}$$

$$\sqrt{1-0.8^2/1^2}$$

$$\sqrt{1-0.64}$$

$$\sqrt{1-0.8^2/1^2}$$

$$\sqrt{1-0.64}$$

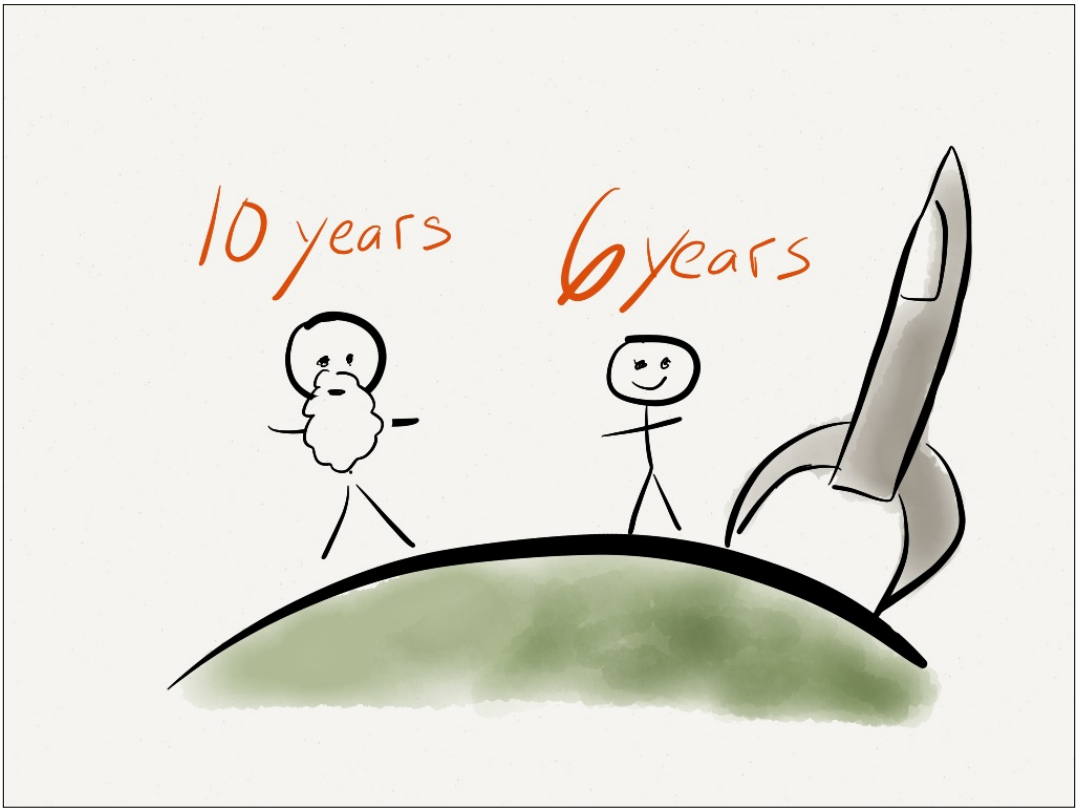
$$\sqrt{0.36}$$

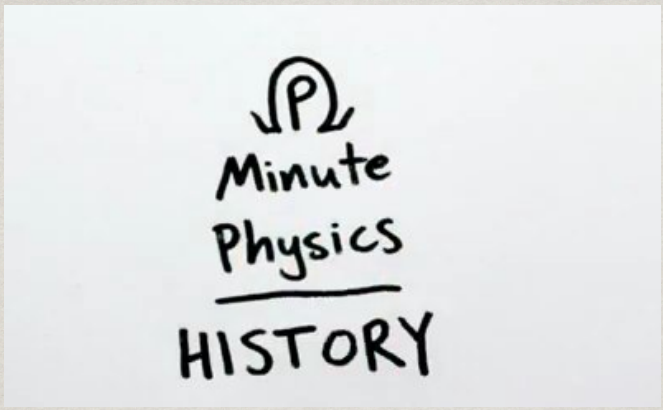
$$\sqrt{1-0.8^2/1^2}$$

$$\sqrt{1-0.64}$$

$$\sqrt{0.36}$$

$$0.6$$

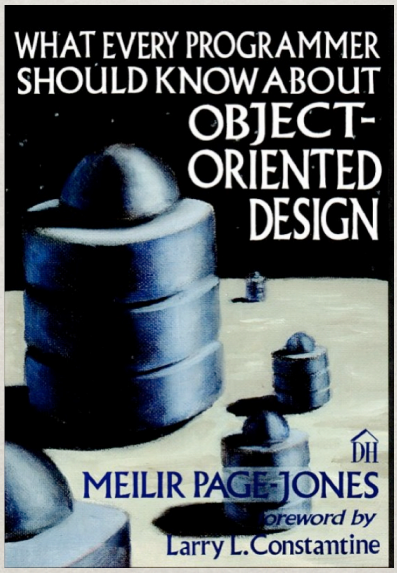




http://www.youtube.com/watch?v=ajhFNcUTJI0

http://bit.ly/special_relativity

CONNASCENCE



Definition

Connascence

- The common birth of two or more at the same time; the production of two or more together.
- That which is born or produced with another
- The act of growing together.

CONNASCENCE

- NAME
- POSITION
- MEANING
- ALGORITHM
- TYPE
- EXECUTION
- TIMING
- VALUE
- IDENTITY

PRINCIPLES

- DEGREE
- LOCALITY
- STABILITY

CONTRANASCENCE

CONNASCENCE

- NAME
- POSITION
- MEANING
- ALGORITHM
- TYPE
- EXECUTION
- TIMING
- VALUE
- IDENTITY

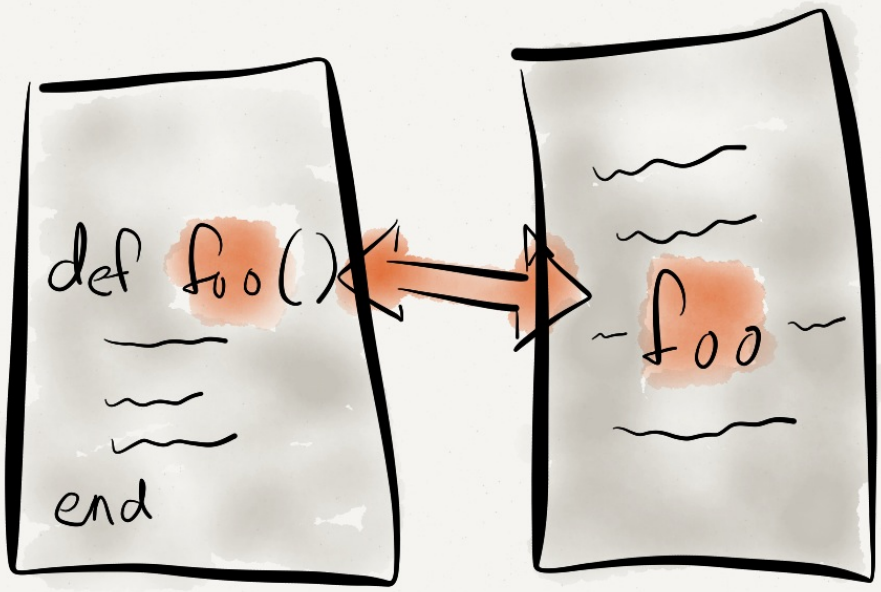
PRINCIPLES

- DEGREE
- LOCALITY
- STABILITY

CONTRANASCENCE

CONNASCENCE OF

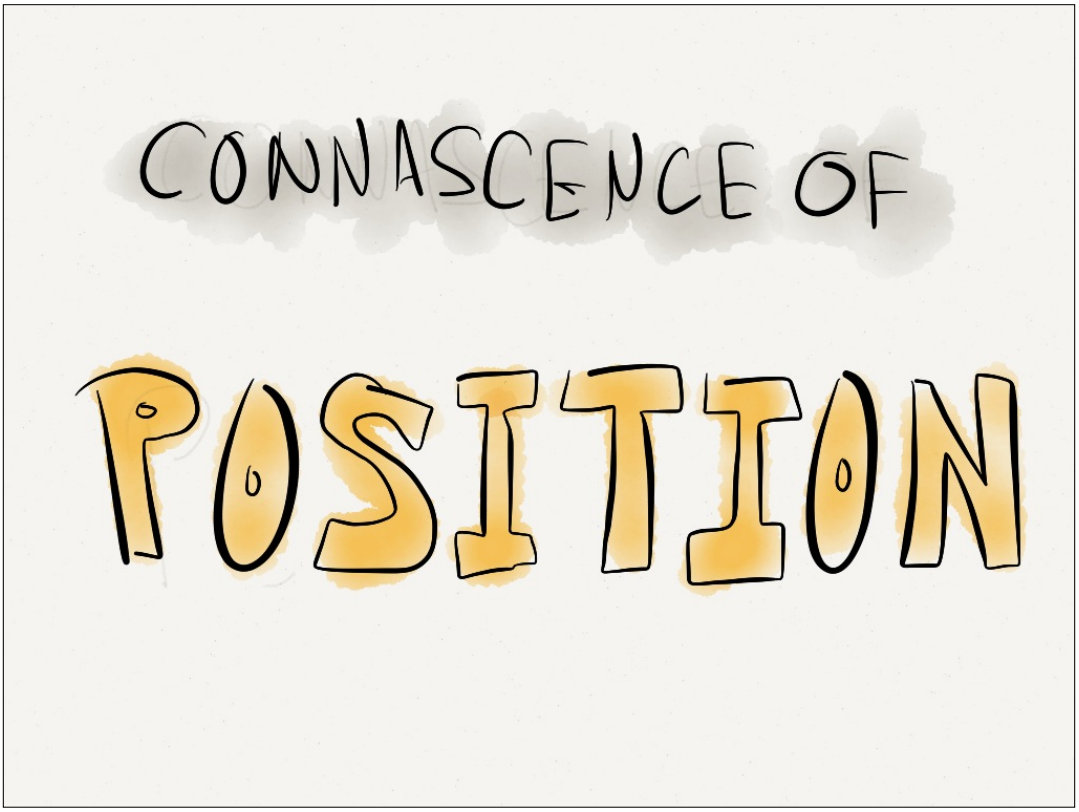
NAME

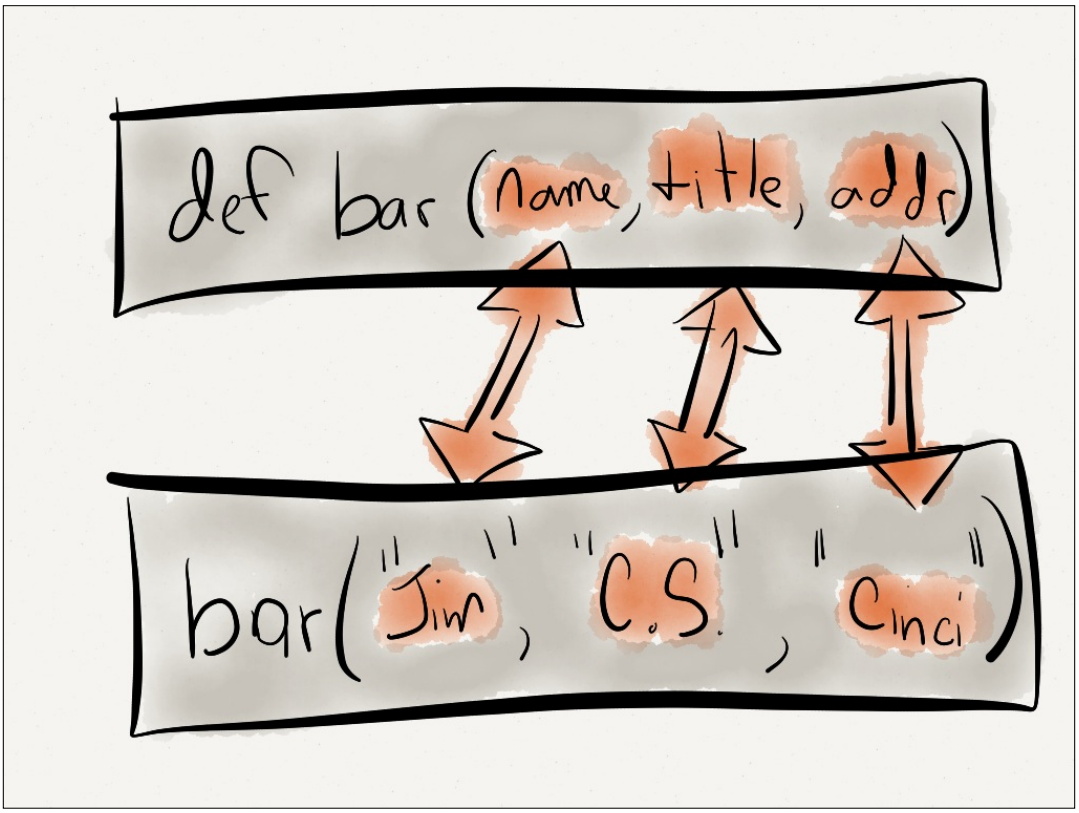


Definition

Connascence of Name

occurs whenever two components must agree on the same name.

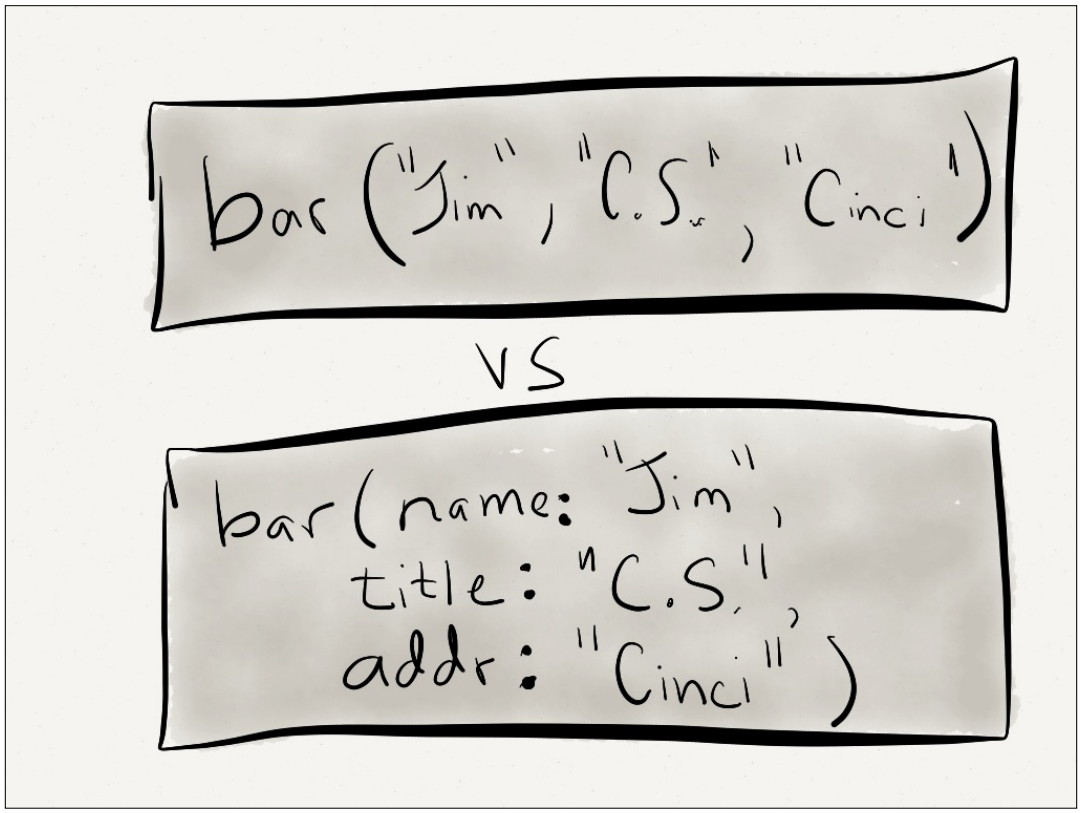


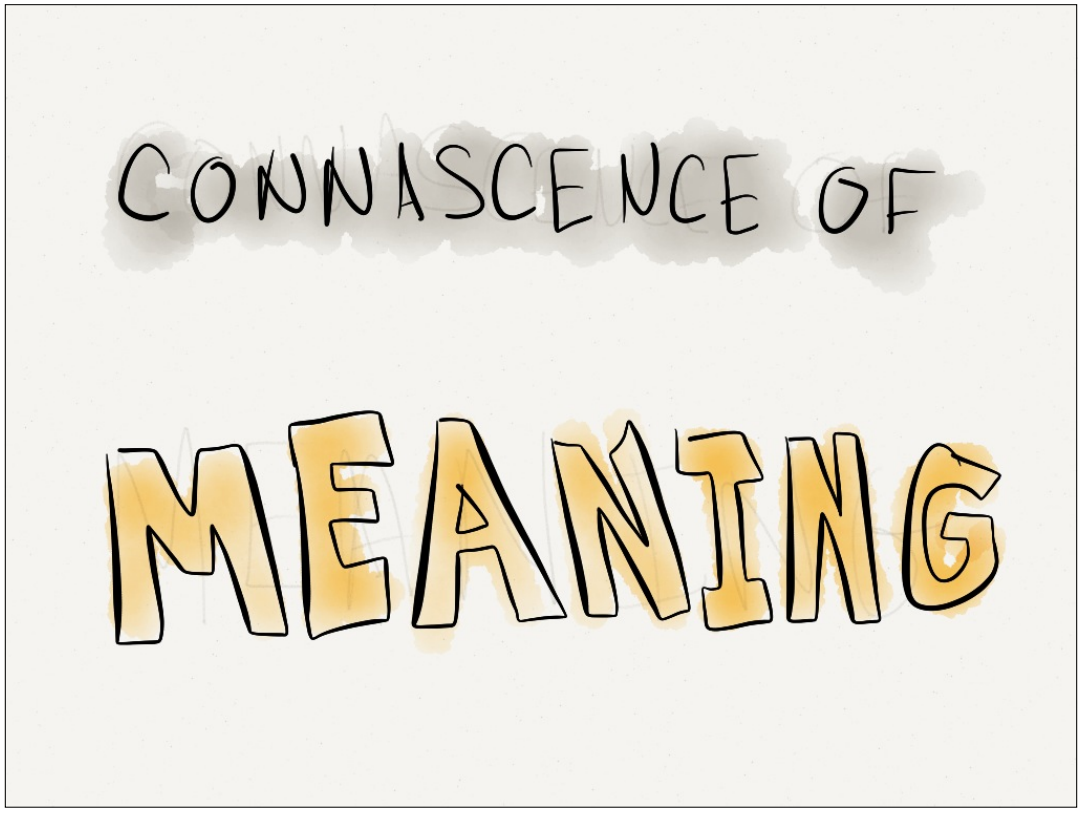


Definition

Connascence of Position

occurs whenever two components must be adjacent or appear in a particular order.


bar("Jim", "C.S.", "Cinci")
VS
bar(name: "Jim",
title: "C.S.",
addr: "Cinci")


CONNASCENCE OF
MEANING

```
...
if ssn == “999-99-9999”
  # No SSN Given
else
  # Handle SSN
end
...
```



```
...
if ssn == MISSING_SSN
  # No SSN Given
else
  # Handle SSN
end
...
```

```
class Phone
  CARRIERS = {
    10              => "Alltel",
    11              => "AT&T",
    12              => "Nextel",
    13              => "Sprint",
    14              => "T-Mobile",
    (WIFI=15)       => "WiFi",
    (OTHER=16)      => "Other Carrier",
    (UK=17)         => "UK Carriers",
    (CANADA=18)     => "Canadian carriers",
  }
end
```



```
<% unless carrier == Phone::WIFI %>
  <% f.text_field :phone_number
<% end %>
```

Definition

Connascence of Meaning

occurs whenever two components must agree on the interpretation of data values.

What Does this Do?

```
Fixnum.instance_methods
```

Which is Which?

```
Fixnum.instance_methods(true)
```

```
Fixnum.instance_methods(false)
```


What Does it Return?

```
User.find(:all, ...)
```

```
User.find(:first, ...)
```

What Does it Return?

```
collection    = User.find(:all, ...)
```

```
single_object = User.find(:first, ...)
```


Which is Which?

```
def User.locate(all_or_first, ...)
  result = find(all_or_first, ...)

  # Did we find anything?
  #
  # if result.nil? ... end
  #   or
  # if result.empty? ... end
end
```

Definition

Control Coupling

occurs when one components passes a piece of information that is intended to control the internal logic of the other.

Warning Signs

Function uses OR in its description

- Returns the methods in the class or class hierarchy
- Finds a single object OR objects
- Reads OR Writes to the disk

Warning Signs

Data that has no intrinsic meaning:

- Symbols
- True / False
- Nil

Warning Signs

Switching Logic

```
def foo(flag)
  if flag
    # use this logic
  else
    # use that logic
  end
  ...
end
```

CONNASCENCE OF ALGORITHM


```
class Page < ActiveRecord::Base

  validates_format_of :title,
    :with => /^([A-Z][a-z]+){2,}$/,
    :message => "must be a WikiName"

  def rendered_content
    text = content || ""
    linked_text =
      text.gsub(/\b([A-Z][a-z]+){2,}\b/) { |name| page_link(name) }
    BlueCloth.new(linked_text).to_html.html_safe
  end

  # ...
end
```

```
class Page < ActiveRecord::Base

  validates_format_of :title,
    :with => /^([A-Z][a-z]+){2,}$/
    :message => "must be a WikiName"

  def rendered_content
    text = content || ""
    linked_text =
      text.gsub(/\b([A-Z][a-z]+){2,}\b/) { |name| page_link(name) }
    BlueCloth.new(linked_text).to_html.html_safe
  end

  # ...
end
```

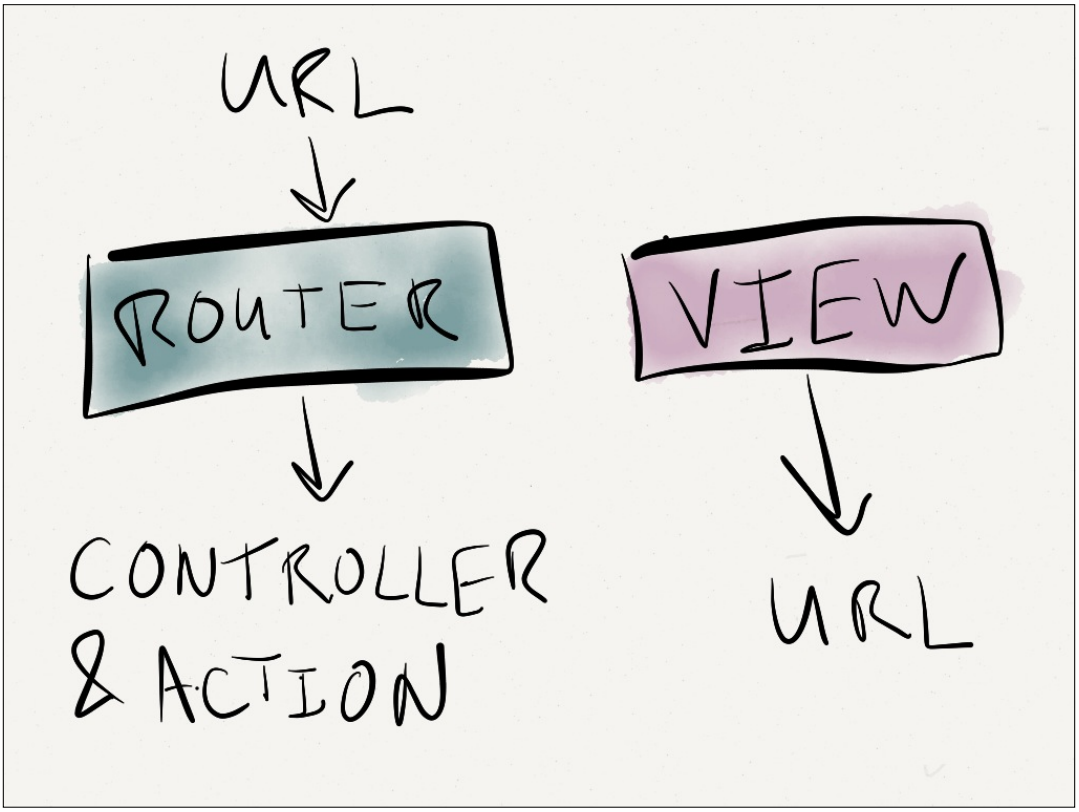


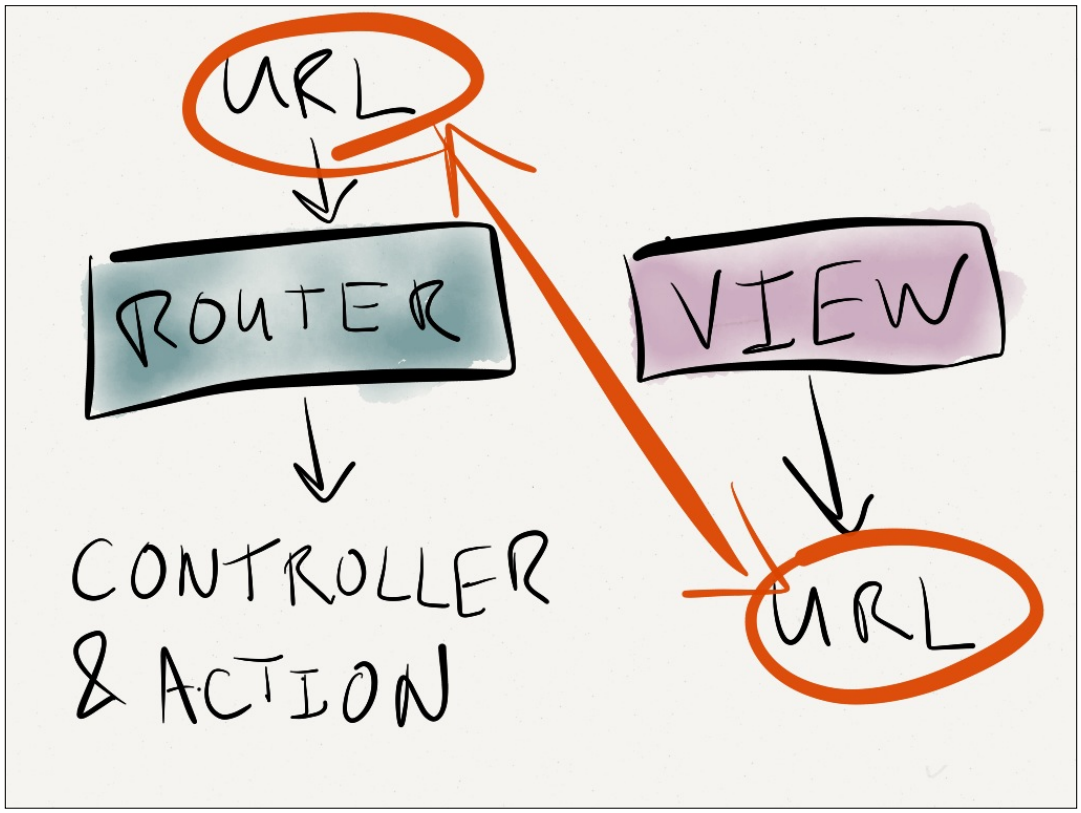
```
class Page < ActiveRecord::Base

  WIKI_RE = /\b([A-Z][a-z]+){2,}\b/
  WIKI_TITLE_RE = /^#{WIKI_RE}$/

  validates_format_of :title, :with => WIKI_TITLE_RE,
    :message => "must be a WikiName"

  def rendered_content
    text = content || ""
    linked_text = text.gsub(WIKI_RE) { |name| page_link(name) }
    BlueCloth.new(linked_text).to_html.html_safe
  end

  # ...
end
```

Definition

Connascence of Algorithm

occurs whenever two components must agree on a particular algorithm.






```
<%= link_to "Create Page", "/pages;new" %>
```

```
<%= link_to "Create Page", "/pages/new" %>
```



```
<%= link_to "Create Page", new_pages_path %>
```

REDUCE DEGREE

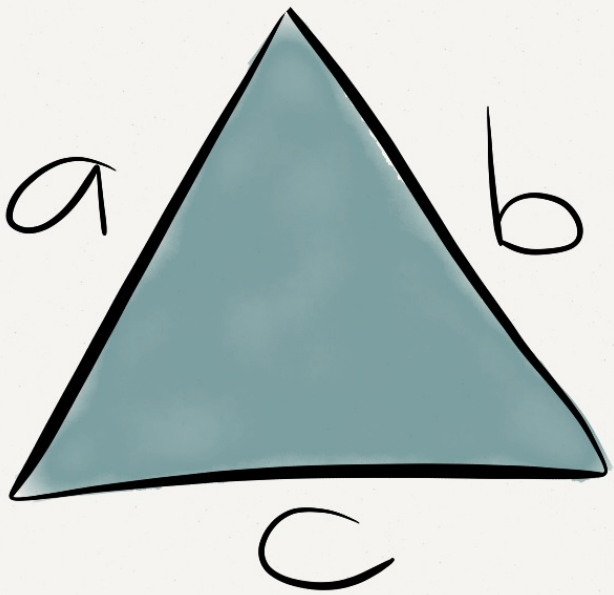
CONNASCENCE OF TYPE

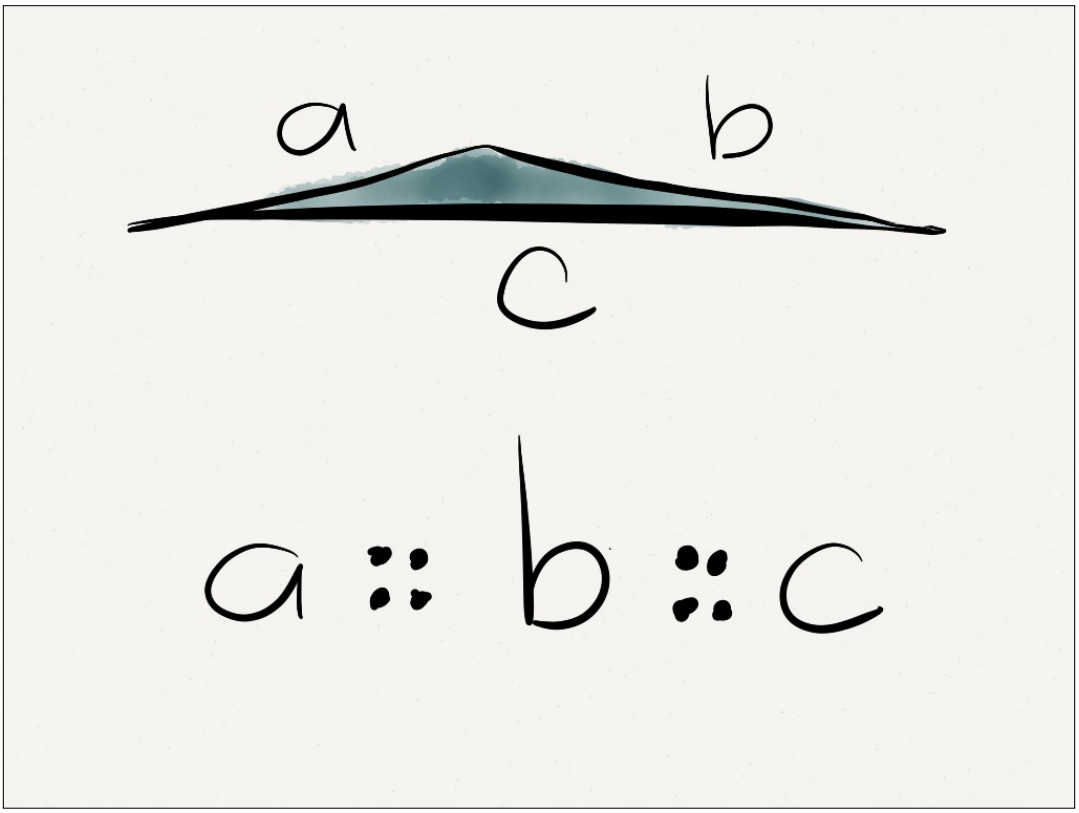
Lets come back to this later ...

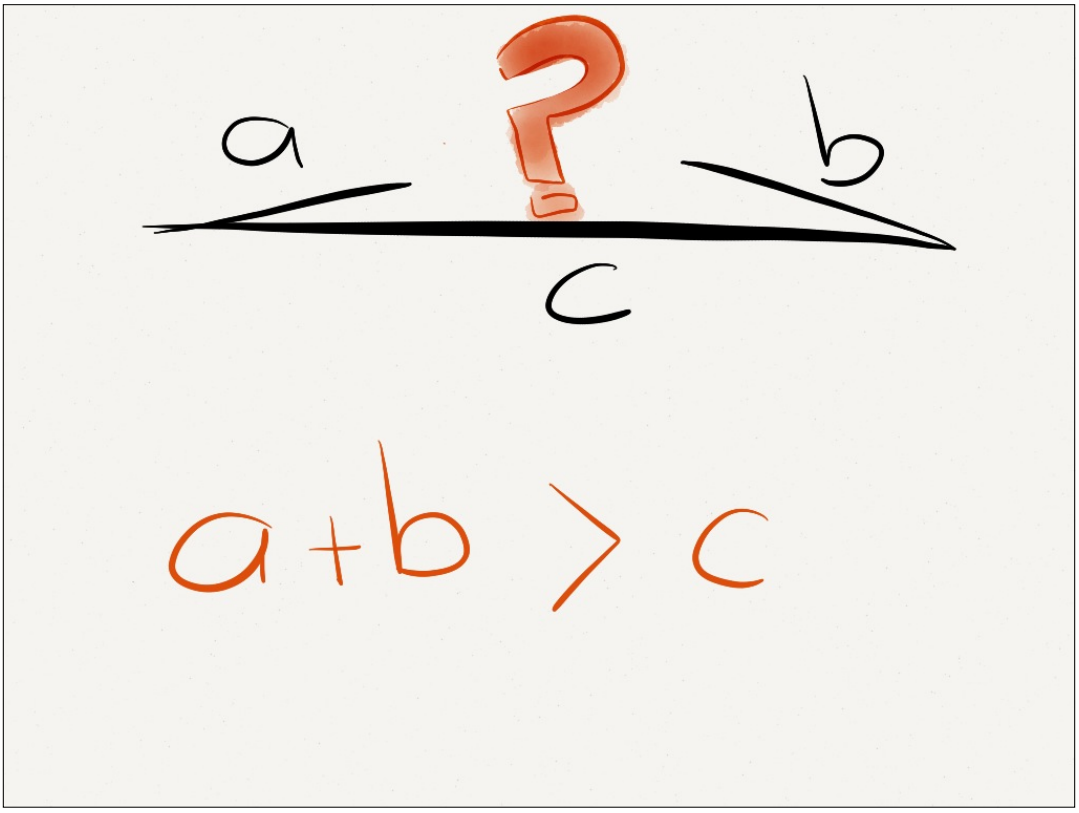
CONNASCENCE OF

VALUE

```
class Triangle
  attr_reader :a, :b, :c

  def initialize(a, b, c)
    @a, @b, @c = a, b, c
  end

  def valid?
    # What relationship exists
    # between a, b, and c?
  end
end
```

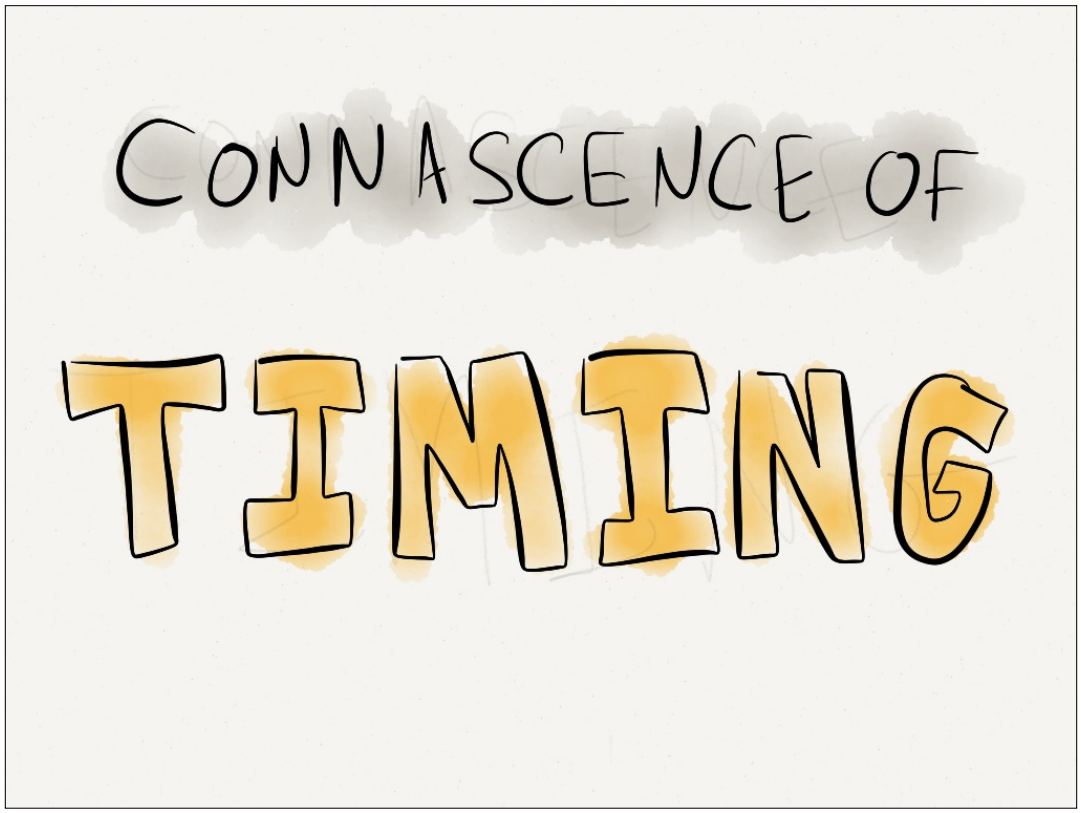

a
b
c
a :: b :: c


a
?
b
c
a+b > c

```
class Triangle
  attr_reader :a, :b, :c

  def initialize(a, b, c)
    @a, @b, @c = a, b, c
  end

  def valid?
    x, y, z = [a, b, c].sort
    x + y > z
  end
end
```


Definition

Connascence of Value
occurs when the values of two components are related.




```
class Account
  attr_reader :balance

  def initialize
    @balance = 0
  end

  def deposit(amount)
    @balance += amount
  end
end
```

```
THREADS = 10
account = Account.new

threads = (0...THREADS).map { |i|
  Thread.new {
    1000.times {
      account.deposit(1)
      sleep(0.001)
    }
  }
}
threads.each { |t| t.join }

puts account.balance
```



```
$ ruby co_timing.rb
10000
$ ruby co_timing.rb
9991
$ ruby co_timing.rb
10000
$ ruby co_timing.rb
9992
$ ruby co_timing.rb
9992
$ ruby co_timing.rb
10000
```

Definition

Connascence of Timing occurs when the timing of execution is important.

Examples

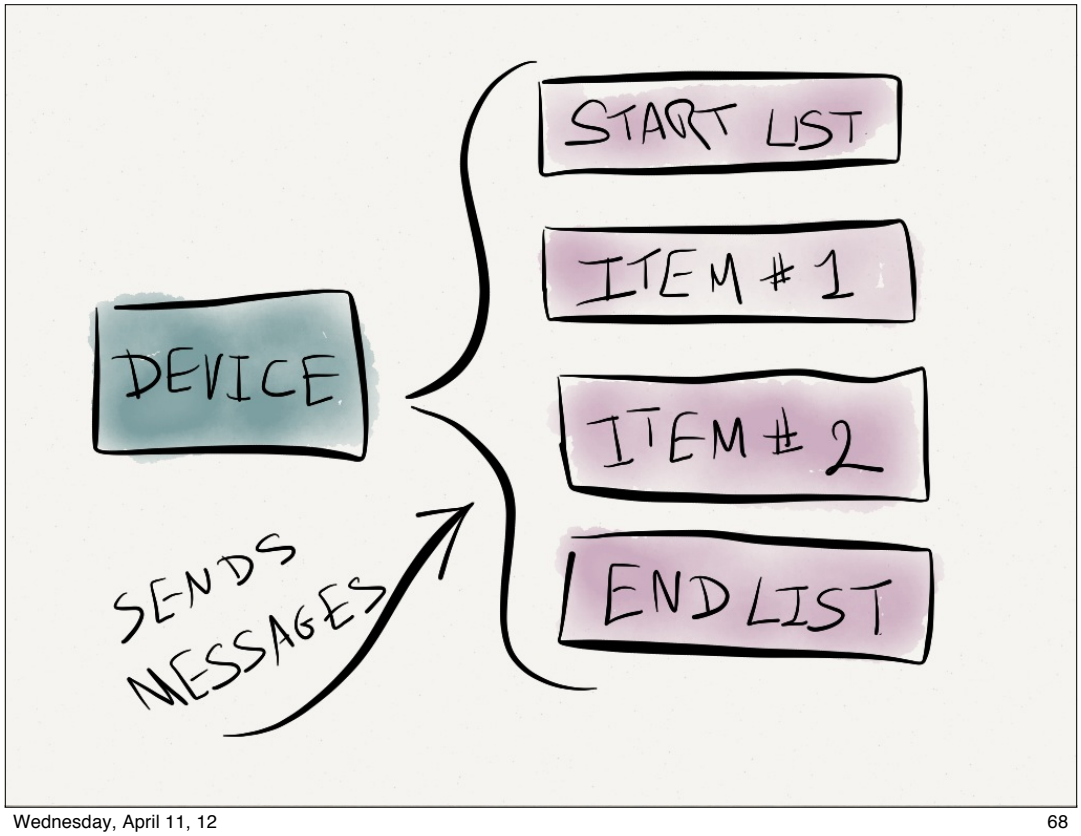
Connascence of Timing

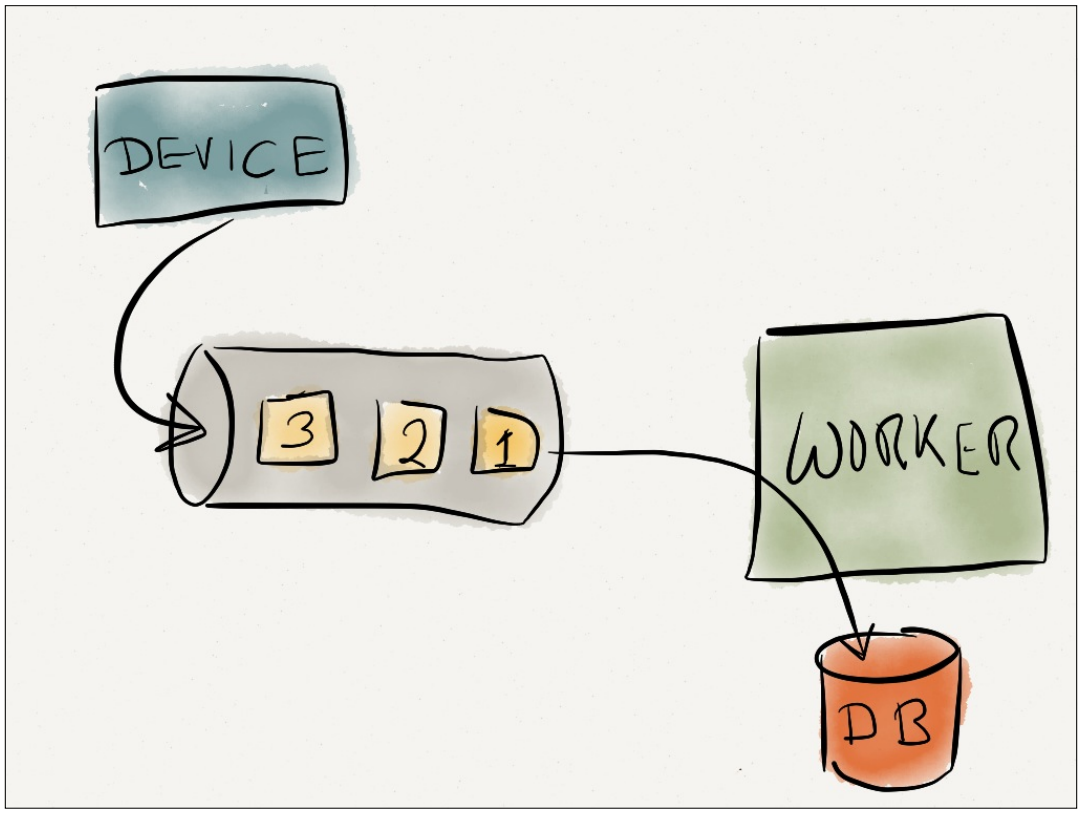
- Race Conditions
- Timeouts

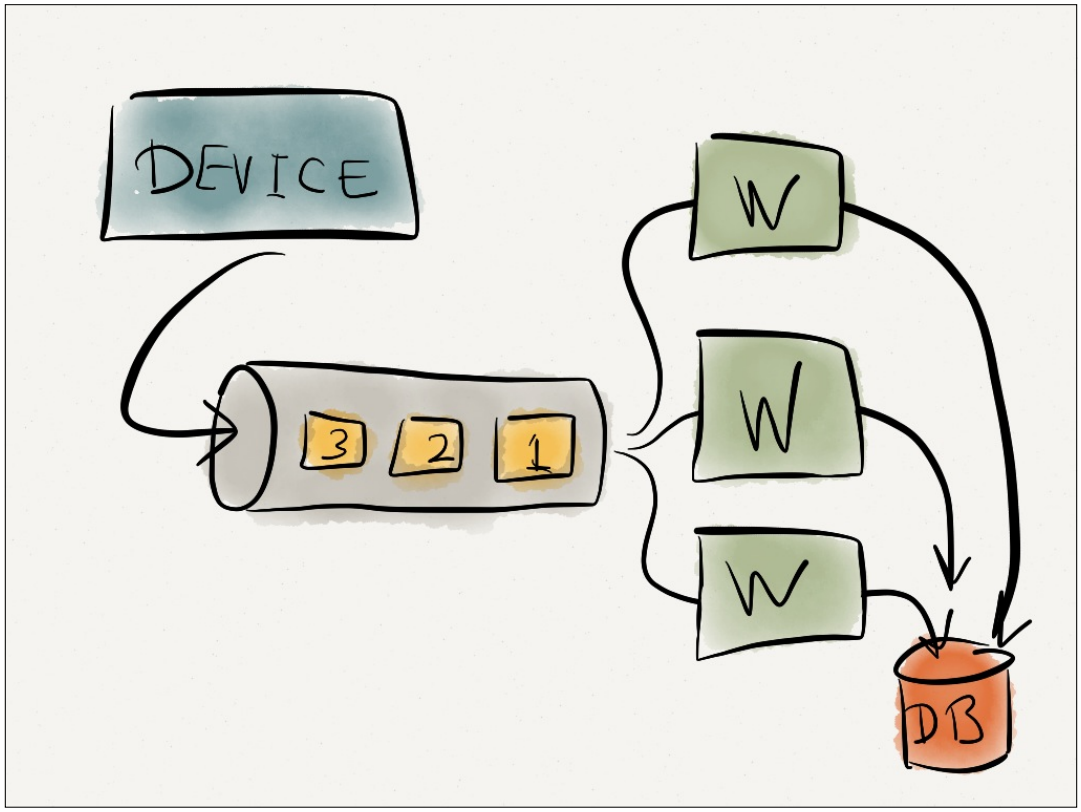
CONNASCENCE OF EXECUTION

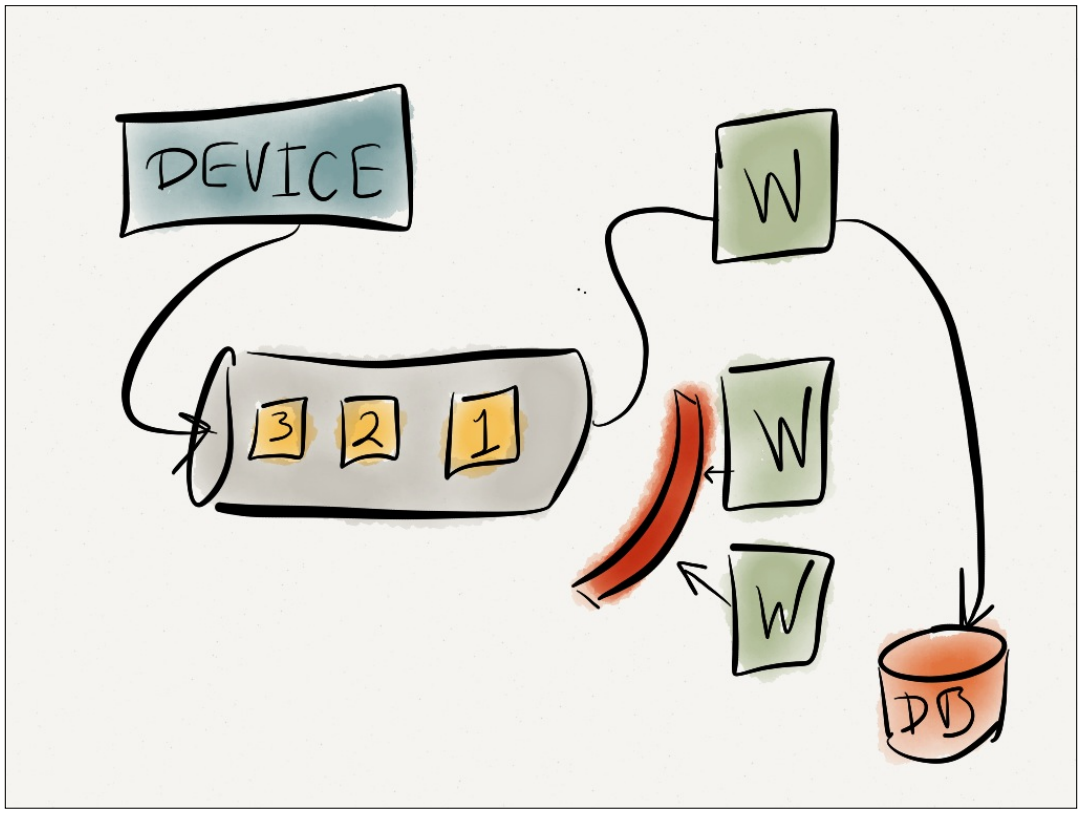

```
receipts.each do |receipt|
  submit_to_edgecase(receipt)
  file_locally(receipt)
  purge(receipt)
end
```

```
receipts.each do |receipt|
  submit_to_edgecase(receipt)
  file_locally(receipt)
  purge(receipt)
end
```


DEVICE
3
2
1
WORKER
DB


DEVICE
3
2
1
W
W
W
DB

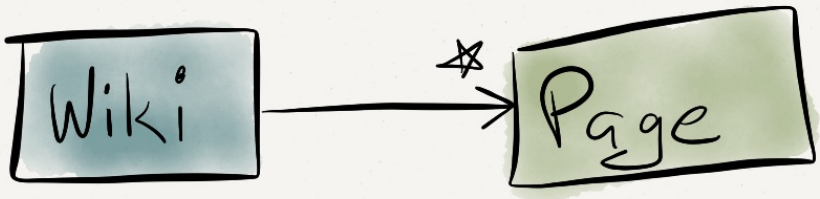


Definition

Connascence of Execution occurs when the *order* of execution of two components is important.

CONNASCENCE OF

IDENTITY

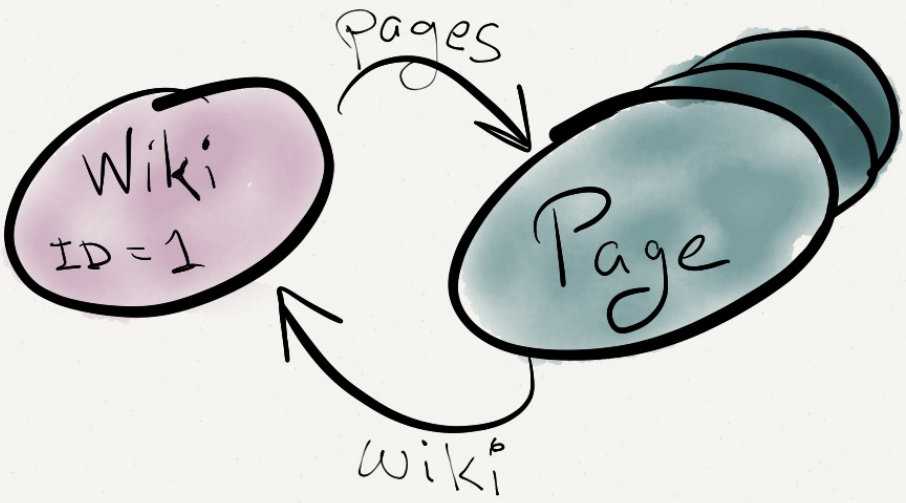


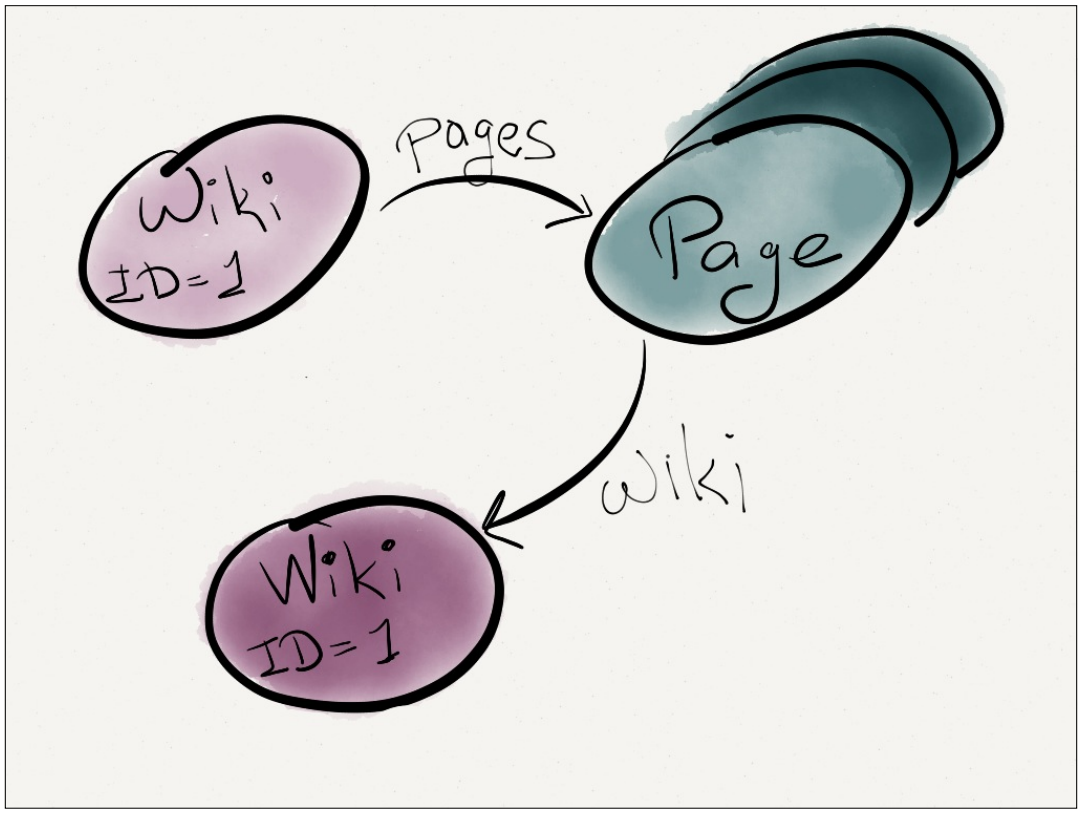
```
class Page < ActiveRecord::Base
  belongs_to :wiki

  # ...

  def make_home_page
    wiki.home = title
    wiki.save
  end
end
```



```
describe "make home page" do
  Given(:wiki) {
    Wiki.create(name: "Wiki", home: "TheOldHomePage")
  }
  Given(:page) {
    wiki.pages.create(title: "TheNewHomePage")
  }

  When { page.make_home_page }

  Then { wiki.home.should == "TheNewHomePage" }
end
```

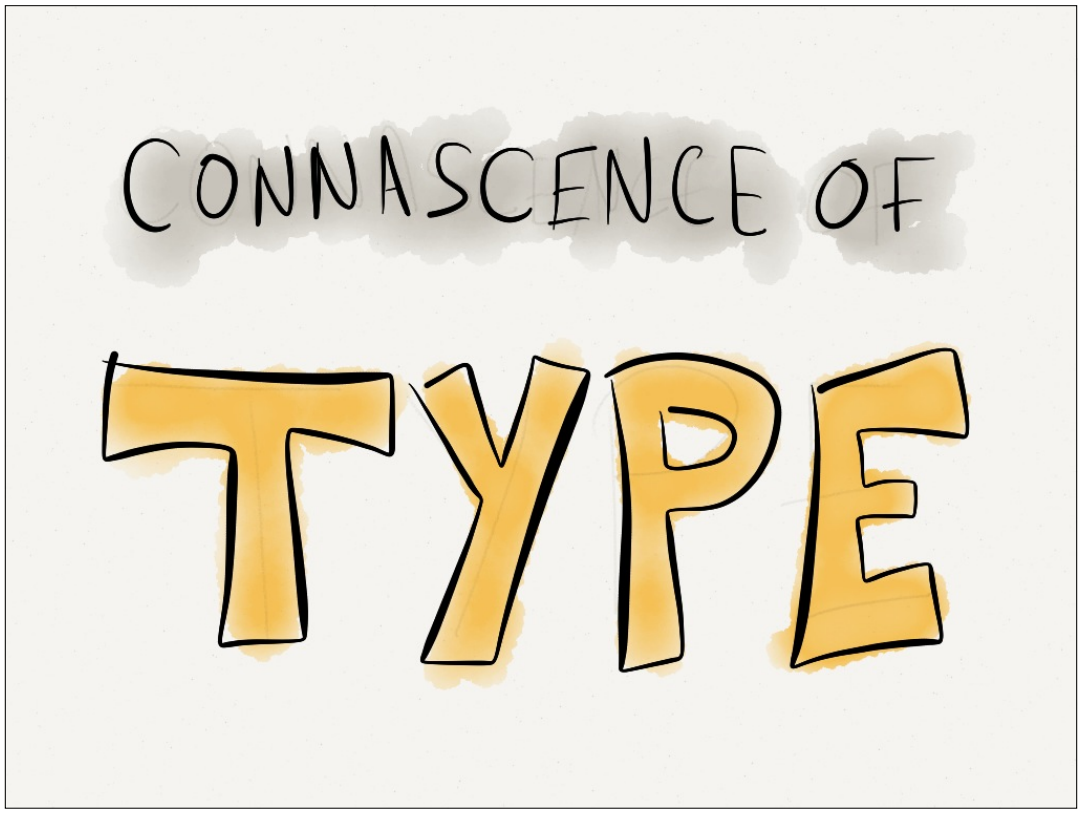
```
1) Page make home page
   Failure/Error:
     Then { wiki.home.should == "TheNewHomePage" }
     expected: "TheNewHomePage"
          got: "TheOldHomePage" (using ==)
```

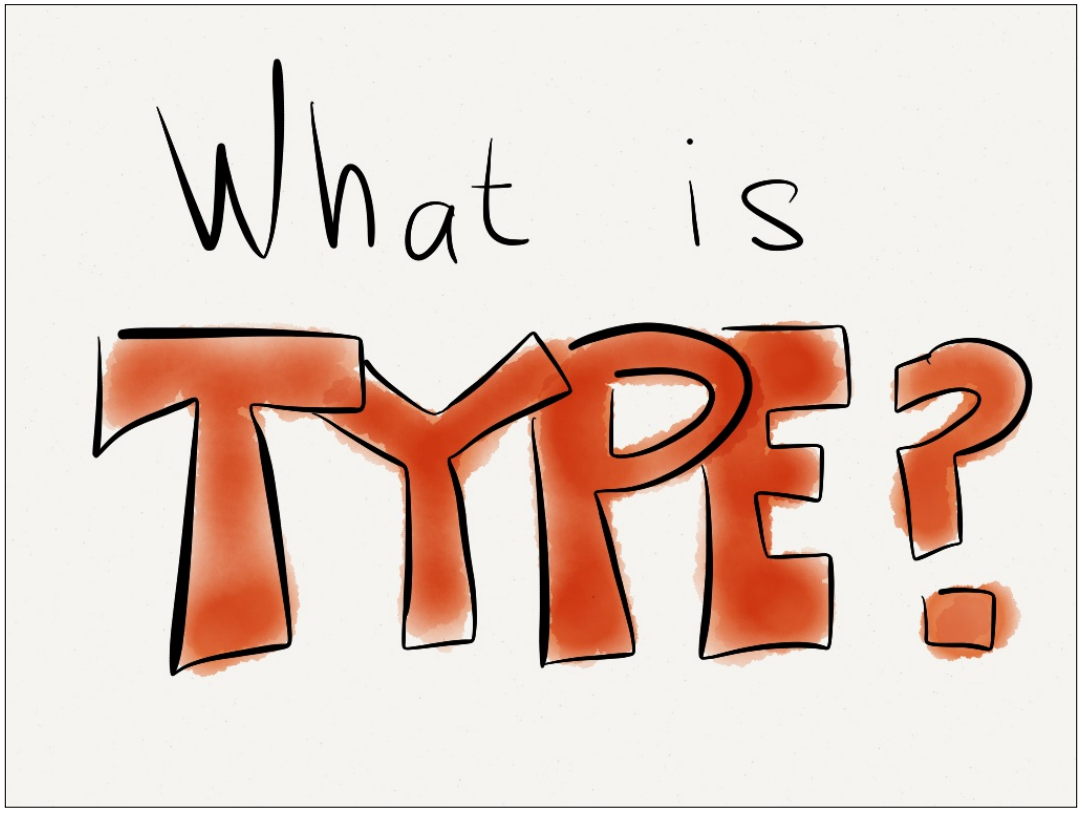



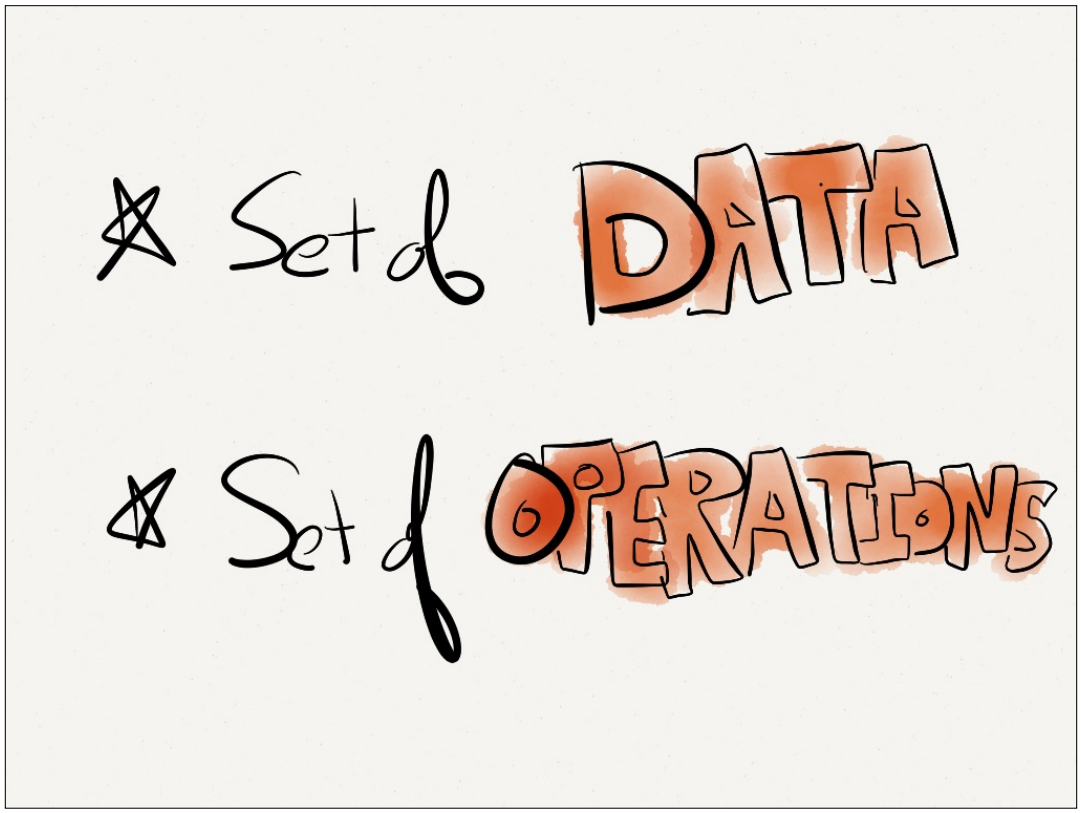


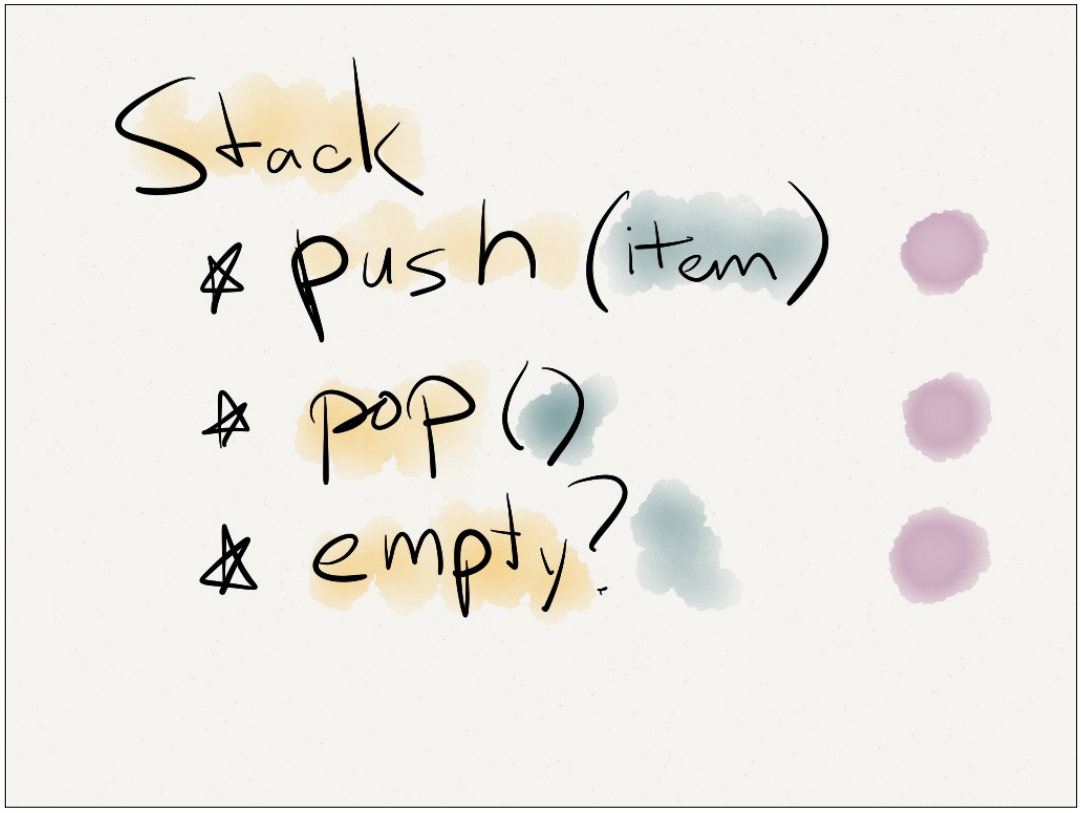
Definition

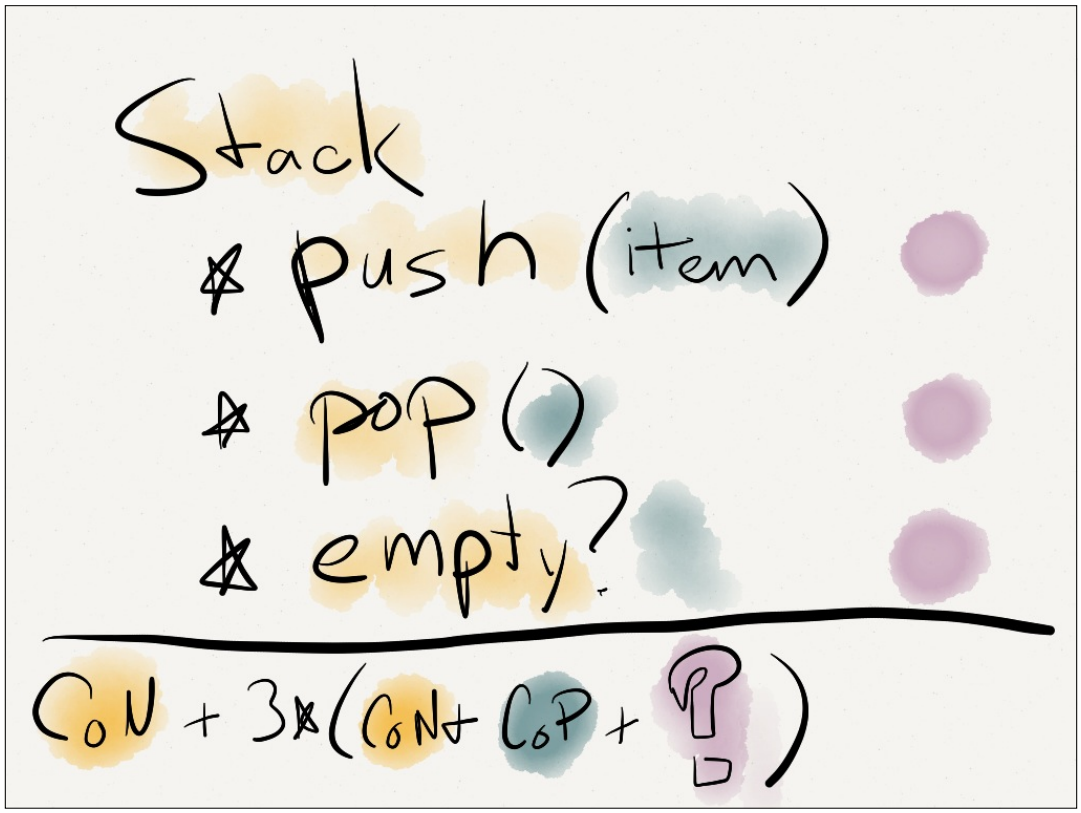
Connascence of Identity
occurs when two components must reference the same object.

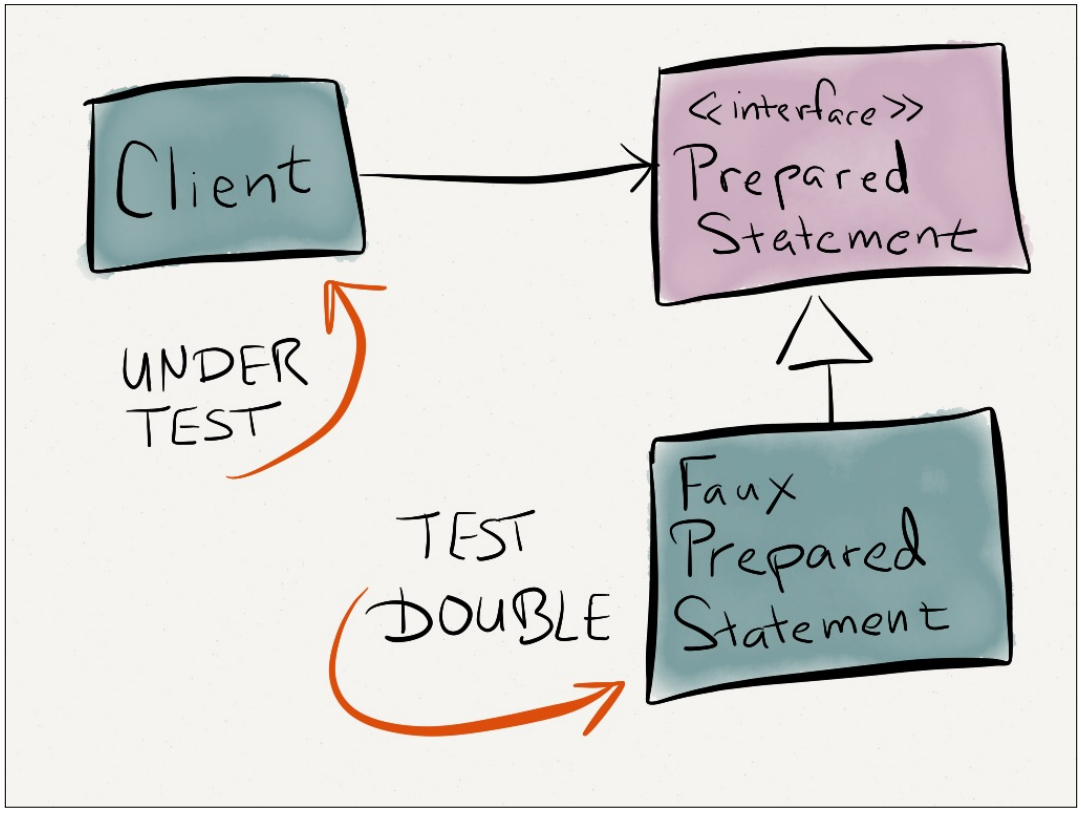

CONNASCENCE OF
TYPE


What is
TYPE?


Set of DATA
Set of OPERATIONS


Stack
push(item)
pop()
empty?


Stack
* push(item)
* pop()
* empty?
C₀N + 3*(C₀N + C₀P +)

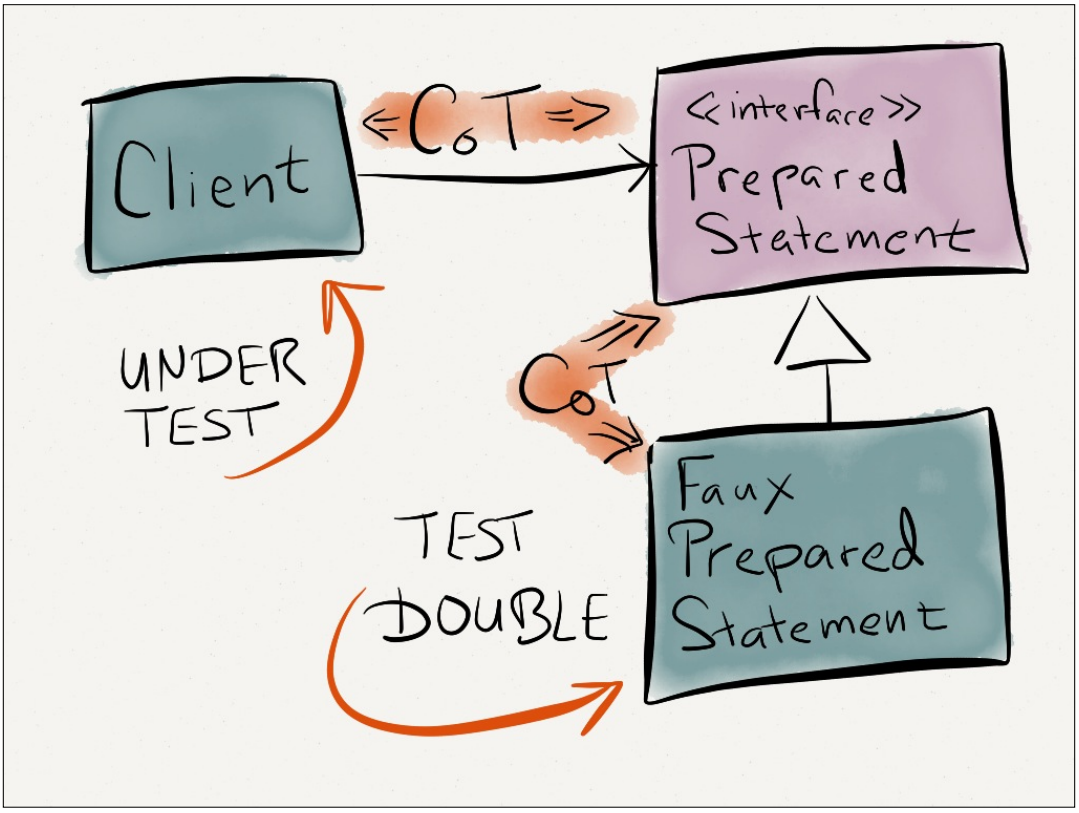

Client
<<interface>>
Prepared Statement
UNDER TEST
TEST DOUBLE
Faux Prepared Statement

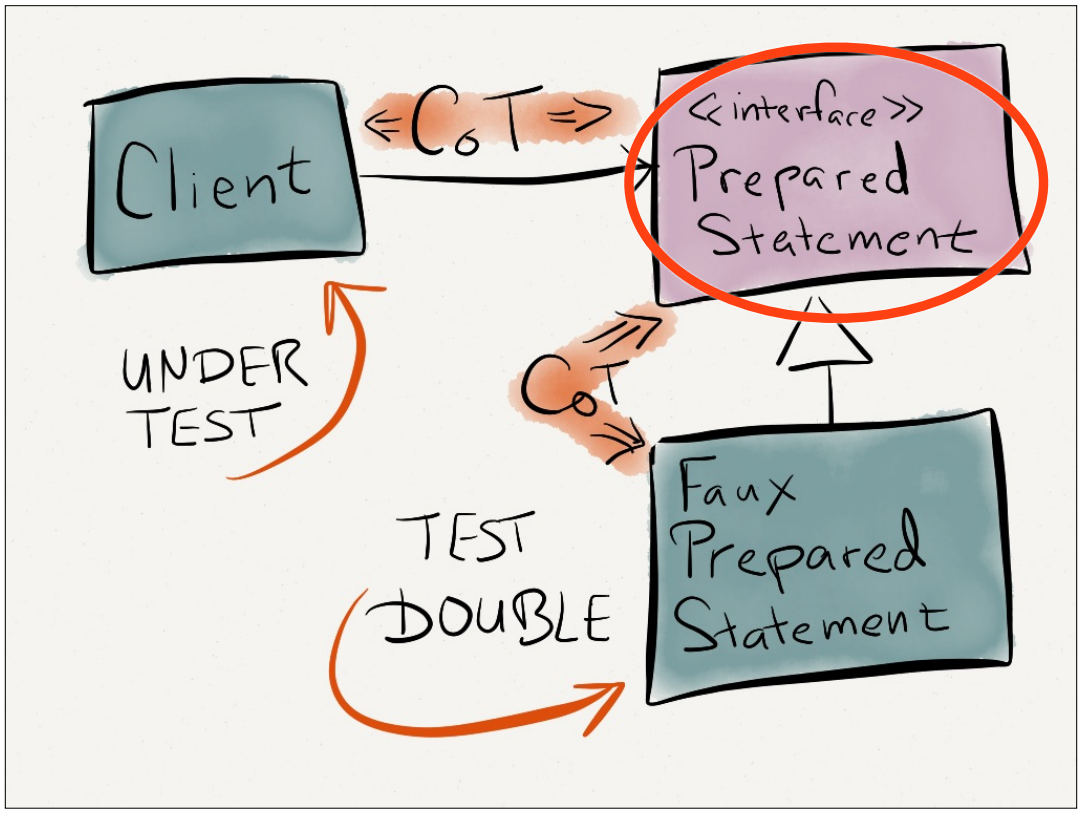
```
class FauxPreparedStatement
    implements java.sql.PreparedStatement
{

    public void setInt(int i, int j) {
        // Faux code here
    }

    public java.sql.ResultSet executeQuery() {
        // Faux code here
    }

    ...
}
```

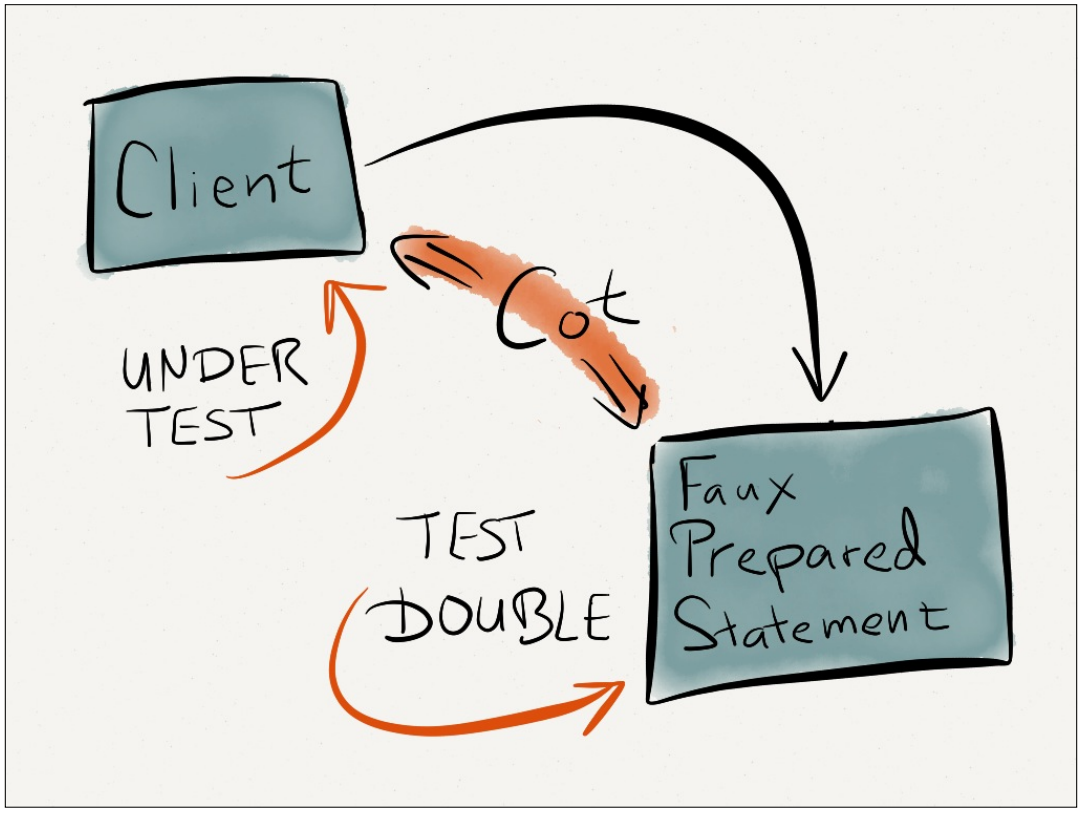


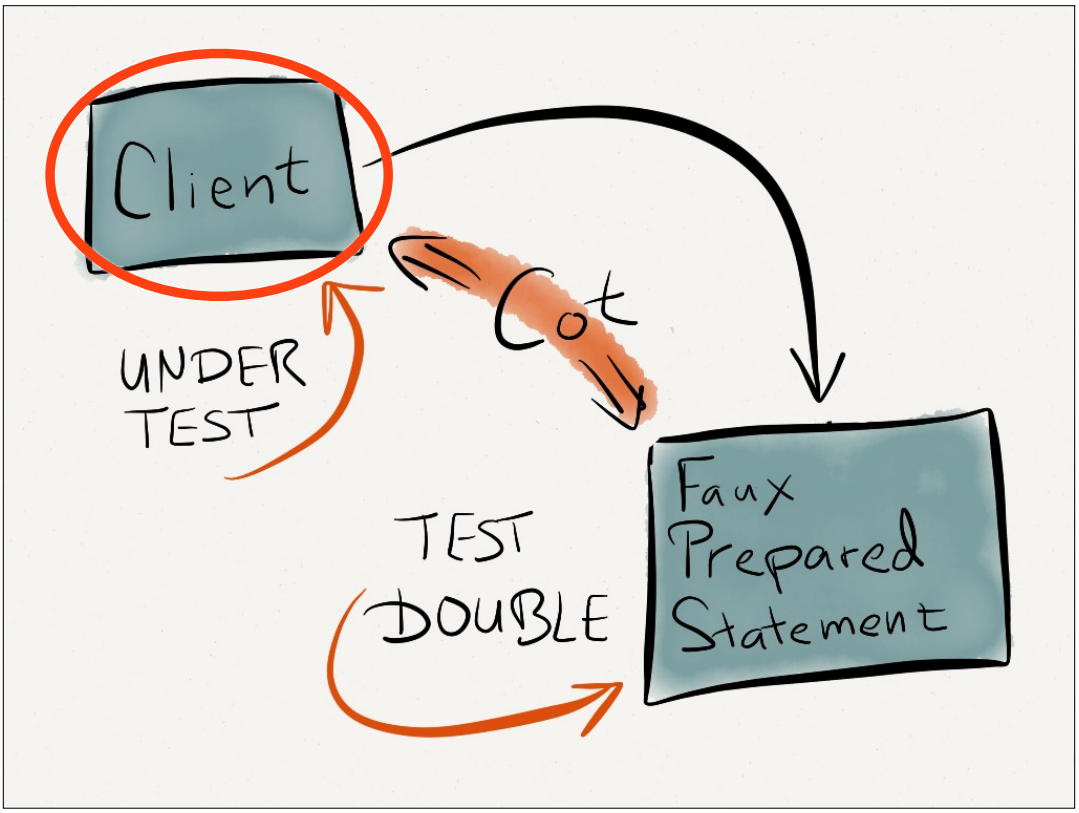
```
package org.onestepback;

class FauxPreparedStatement implements java.sql.PreparedStatement {

    public void setInt(int i, int j) {
        // Faux code here
    }

    public java.sql.ResultSet executeQuery() {
        // Faux code here
    }

    public <T> T unwrap(java.lang.Class<T> klass) { return null; }
    public boolean isWrapperFor(java.lang.Class klass) { return false; }
    public void close() { }
    public int getMaxFieldSize() { return 0; }
    public void setMaxFieldSize(int i) { }
    public void setMaxRows(int i) { }
    public int getMaxRows() { return 0; }
    public void setEscapeProcessing(boolean bool) { }
    public int getQueryTimeout() { return 0; }
    public void setQueryTimeout(int timeout) { }
    public void cancel() { }
    public void clearWarnings() { }
    public java.sql.SQLWarning getWarnings() { return null; }
    public void setCursorName(String name) { }
    public java.sql.ResultSet getResultSet() { return null; }
    public int getUpdateCount() { return 0; }
    public int getFetchDirection() { return 0; }
    public void setFetchDirection(int i) { }
    public int getFetchSize() { return 0; }
    public void setFetchSize(int i) { }
    public int getFetchSize(int i) { return 0; }
    public int getResultSetConcurrency() { return 0; }
    public int getResultSetType() { return 0; }
    public java.sql.Connection getConnection() { return null; }
    public java.sql.ResultSet getGeneratedKeys() { return null; }
    public boolean getMoreResults() { return false; }
    public boolean getMoreResults(int i) { return false; }
    public int getResultSetHoldability() { return 0; }
    public boolean isClosed() { return false; }
    public void setPoolable(boolean bool) { }
    public boolean isPoolable() { return false; }
    public boolean execute(String sql) { return false; }
    public boolean execute(String sql, int param) { return false;  }
    public boolean execute(String sql, int params[]) { return false; }
    public boolean execute(String sql, String params[]) { return false; }
    public boolean execute(String sql) { return false; }
    public void clearBatch() { }
    public int[] executeBatch() { return null; }
    public java.sql.ResultSet executeQuery() { return null; }
    public java.sql.ResultSet executeQuery(String sql) { return null; }
    public int executeUpdate() { return 0; }
    public int executeUpdate(String sql) { return 0; }
    public int executeUpdate(String sql, int param) { return 0; }
    public int executeUpdate(String sql, int[] params) { return 0; }
    public int executeUpdate(String sql, String[] params) { return 0; }
    public void setNull(int i, int j) { }
    public void setBoolean(int i, boolean bool) { }
    public void setByte(int i, byte b) { }
    public void setShort(int i, short s) { }
    public void setLong(int i, long l) { }
    public void setFloat(int i, float f) { }
    public void setDouble(int i, double d) { }
    public void setBigDecimal(int i, java.math.BigDecimal decimal) { }
    public void setString(int i, String string) { }
    public void setBytes(int i, byte[] bytes) { }
    public void clearParameters() { }
    public void addBatch() { }
    public void addBatch(String string) { }
    public void setRef(int i, java.sql.Ref ref) { }
    public void setArray(int i, java.sql.Array array) { }
    public java.sql.ResultSetMetaData getMetaData() { return null; }
    public void setDate(int i, java.sql.Date date) { }
    public void setDate(int i, java.sql.Date date, java.util.Calendar cal) { }
    public void setTime(int i, java.sql.Time time) { }
    public void setTime(int i, java.sql.Time time, java.util.Calendar cal) { }
    public void setTimestamp(int i, java.sql.Timestamp ts) { }
    public void setTimestamp(int i, java.sql.Timestamp ts, java.util.Calendar cal) { }
    public void setNull(int i, int j, String s) { }
    public void setURL(int i, java.net.URL url) { }
    public java.sql.ParameterMetaData getParameterMetaData() { return null; }
    public void setRowId(int i, java.sql.RowId rowId) { }
    public void setNString(int i, String s) { }
    public void setSQLXML(int i, java.sql.SQLXML sqlXml) { }
    public void setObject(int i, Object obj) { }
    public void setObject(int i, Object obj, int j) { }
    public void setObject(int i, Object obj, int j, int k) { }
    public void setAsciiStream(int i, java.io.InputStream stream) { }
    public void setAsciiStream(int i, java.io.InputStream stream, long l) { }
    public void setAsciiStream(int i, java.io.InputStream stream, int j) { }
    public void setUnicodeStream(int i, java.io.InputStream stream, int j) { }
    public void setBinaryStream(int i, java.io.InputStream stream) { }
    public void setBinaryStream(int i, java.io.InputStream stream, int j) { }
    public void setBinaryStream(int i, java.io.InputStream stream, long l) { }
    public void setCharacterStream(int i, java.io.Reader reader) { }
    public void setCharacterStream(int i, java.io.Reader reader, long l) { }
    public void setCharacterStream(int i, java.io.InputStream stream, long l) { }
    public void setCharacterStream(int i, java.io.Reader reader) { }
    public void setNCharacterStream(int i, java.io.Reader reader) { }
    public void setNCharacterStream(int i, java.io.Reader reader, long l) { }
    public void setClob(int i, java.sql.Clob clob) { }
    public void setClob(int i, java.io.Reader reader) { }
    public void setClob(int i, java.io.Reader reader, long l) { }
    public void setBlob(int i, java.sql.Blob blob) { }
    public void setBlob(int i, java.io.InputStream stream) { }
    public void setBlob(int i, java.io.InputStream stream, long l) { }
    public void setNClob(int i, java.io.Reader reader, long l) { }
    public void setNClob(int i, java.sql.NClob nclob) { }
    public void setNClob(int i, java.io.Reader reader) { }
}
```

```
class FauxPreparedStatement
  def set(index, object)
    # Faux code here
  end

  def executeQuery
    # Faux code here
  end
end
```

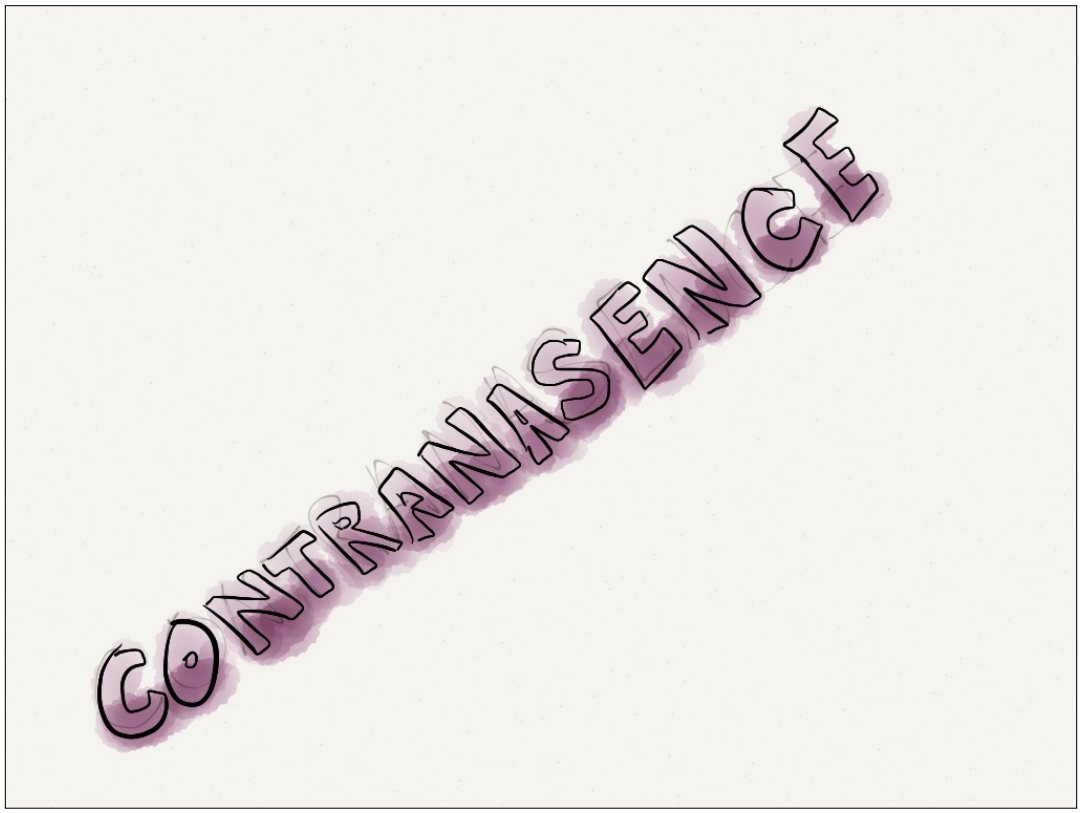

Client
CUT
<<interface>>
Prepared
Statement
UNDER
TEST
COT
Faux
Prepared
Statement
TEST
DOUBLE

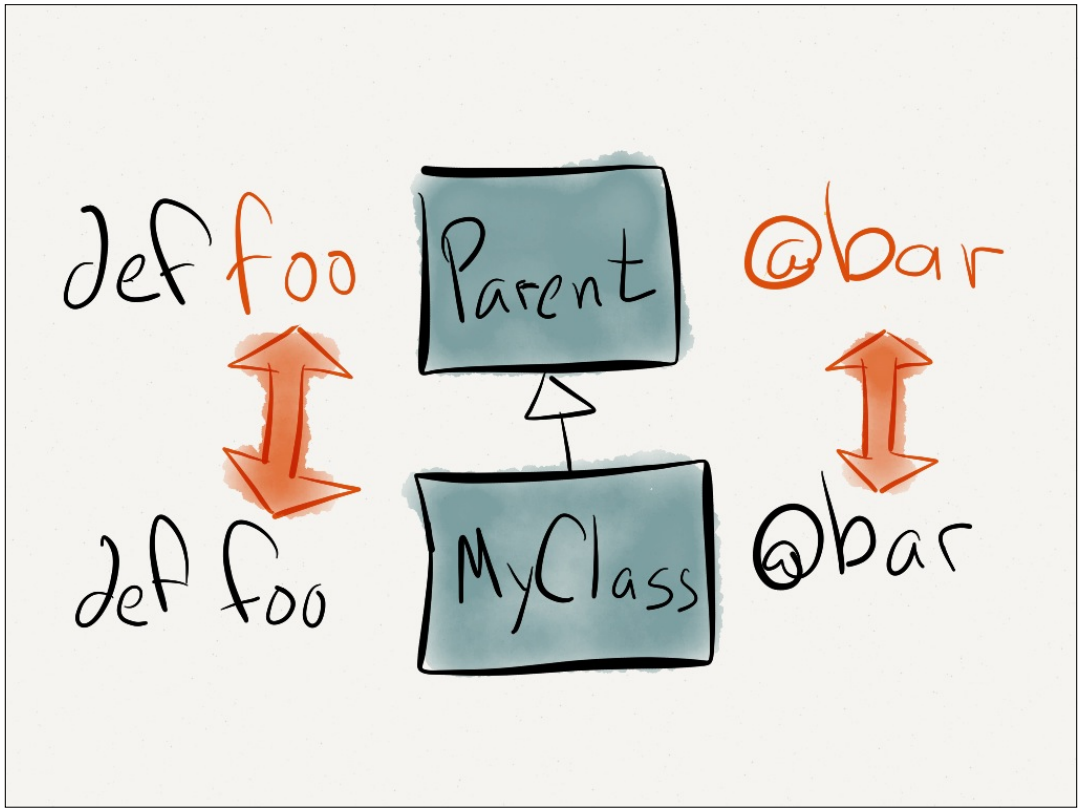

Client
Cot
UNDER
TEST
Faux
Prepared
Statement
TEST
DOUBLE

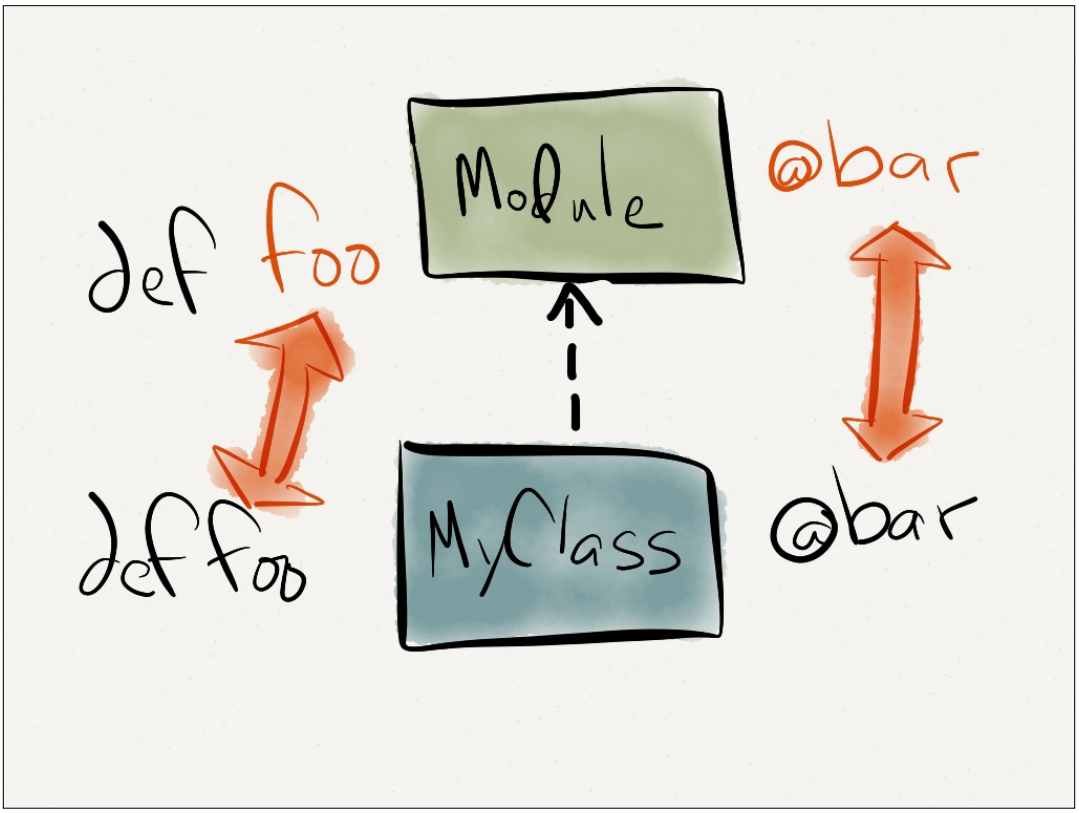


Definition

Connascence of Type
occurs whenever two components must agree on the same type.


CONTRANASENCE


def foo
Parent
@bar
def foo
MyClass
@bar



Definition

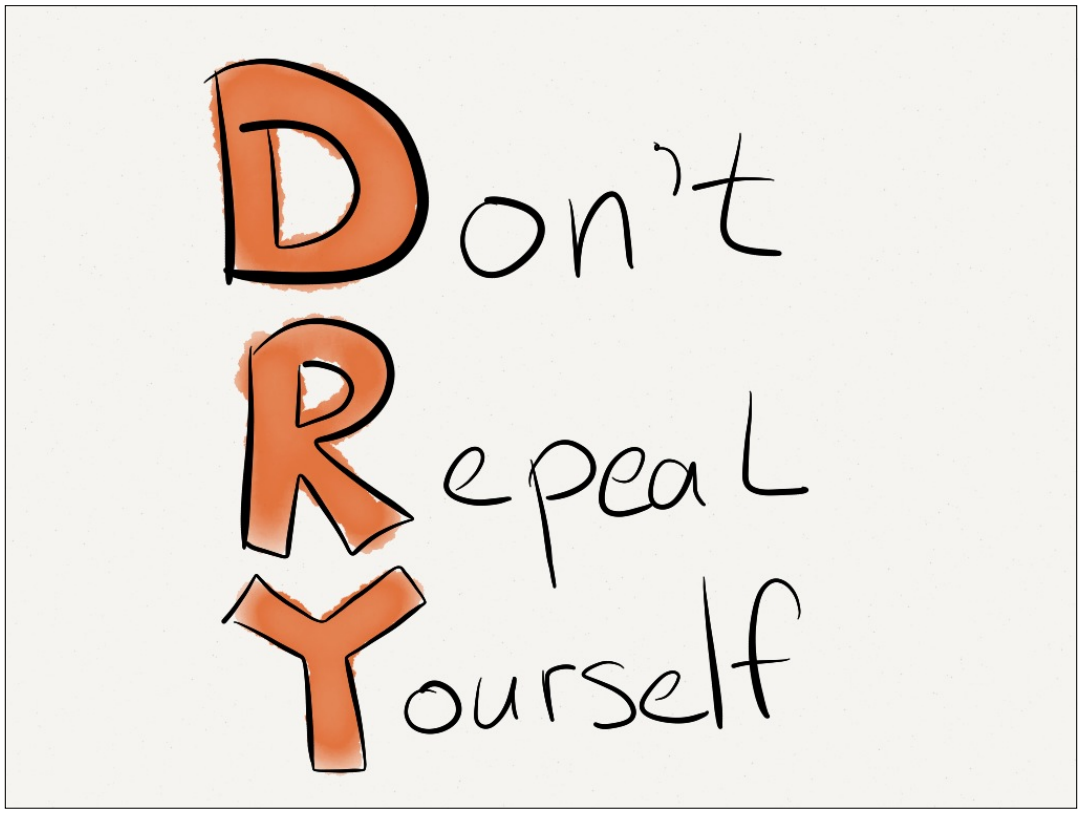
Contranascence
occurs when two components must agree on *different* names.

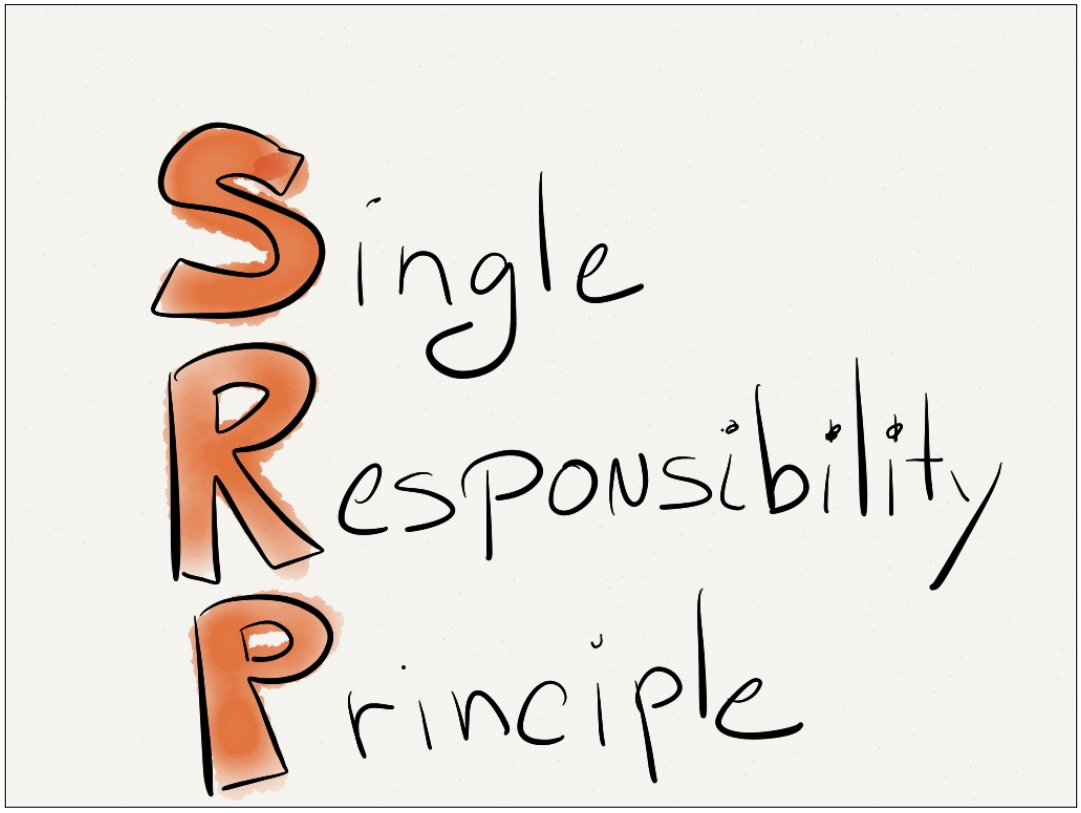
Reducing Contranascence

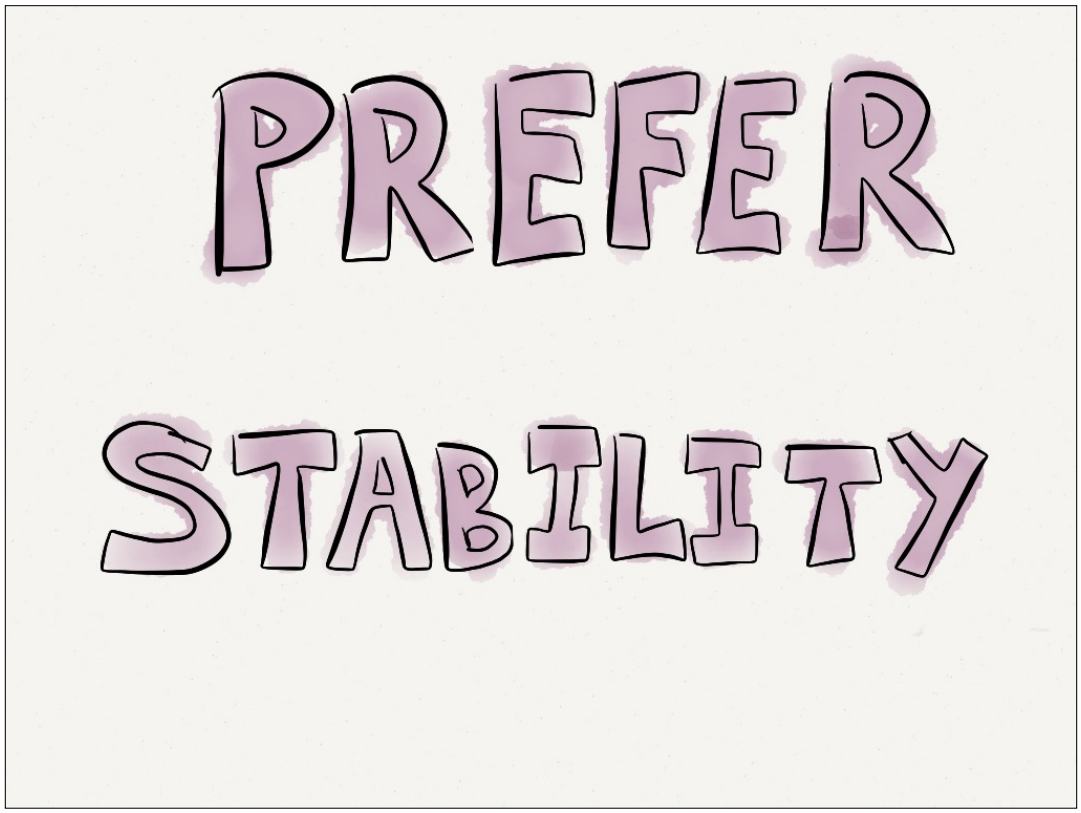
- Namespace
- Delegation VS Inheritance

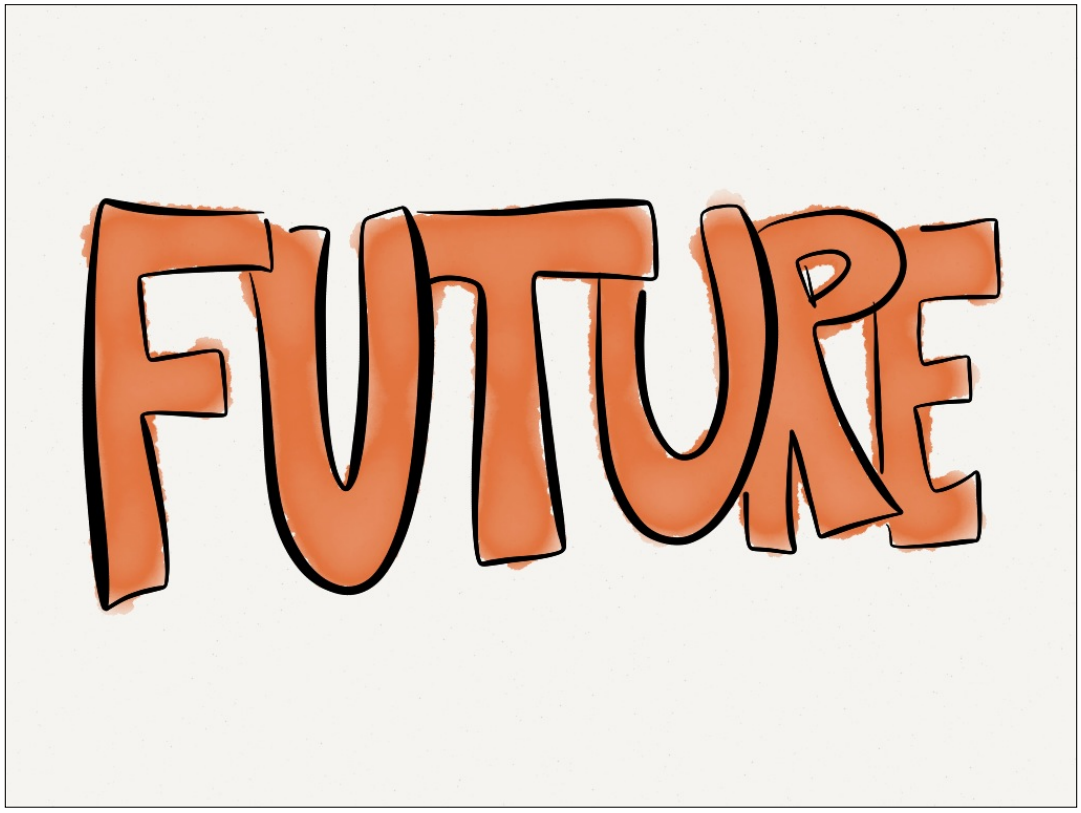
REDUCE DEGREE

INCREASE LOCALITY


Don't
Repeat
Yourself


Single
Responsibility
Principle


PREFER
STABILITY


FUTURE

Questions?

Jim Weirich
Chief Scientist
Edge Case
@jimweirich