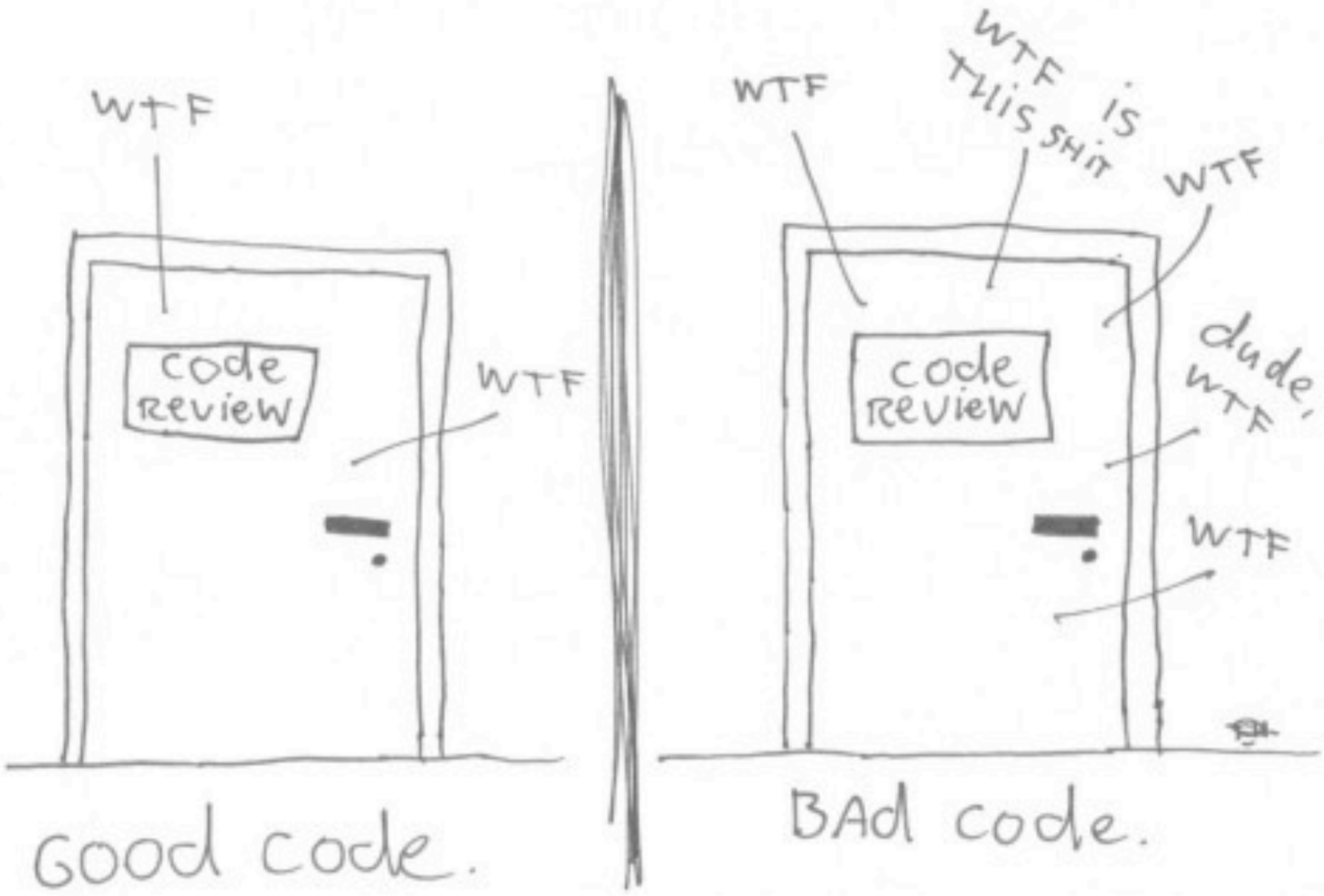
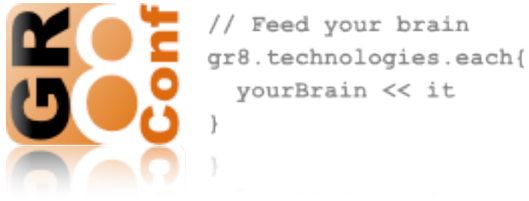
The only valid measurement of code quality: WTFs/minute



Keeping your Grails Code Clean

GR8Conf Europe 2012

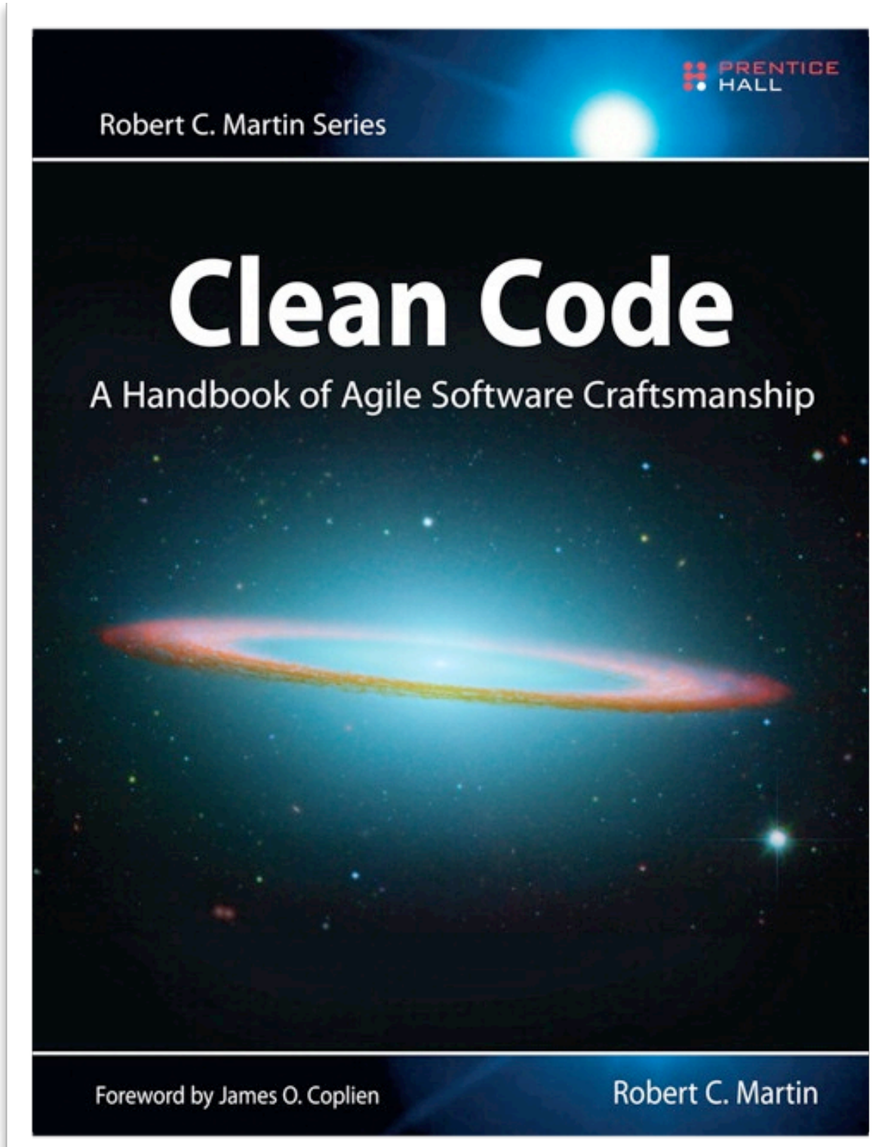
Andreas Arledal
Findout Technologies



What we will cover

- What Clean code is and why is Clean code important?
- Some of the principles of clean code.
- Testable code.
- Grails conventions.
- Tools.

James Duncan Davidson


PRENTICE HALL
Robert C. Martin Series
Clean Code
A Handbook of Agile Software Craftsmanship
Foreword by James O. Coplien
Robert C. Martin

Quiz

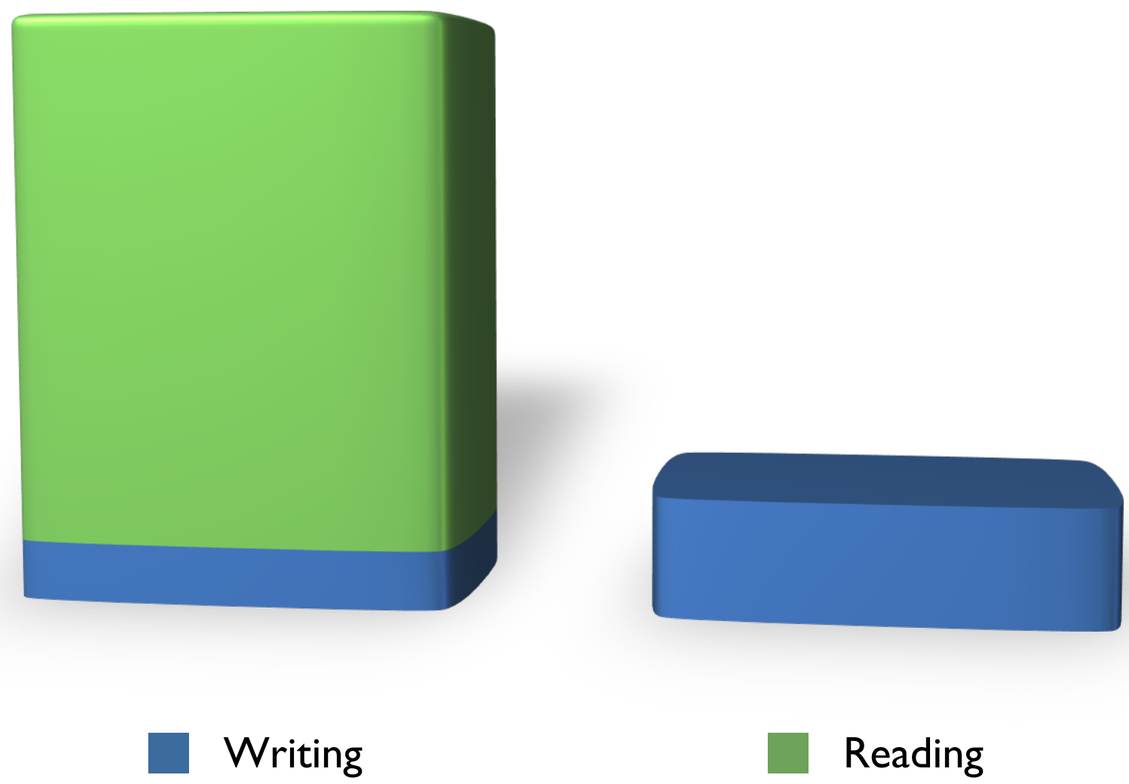
How much of our time do we spend on actually typing code?

Quiz

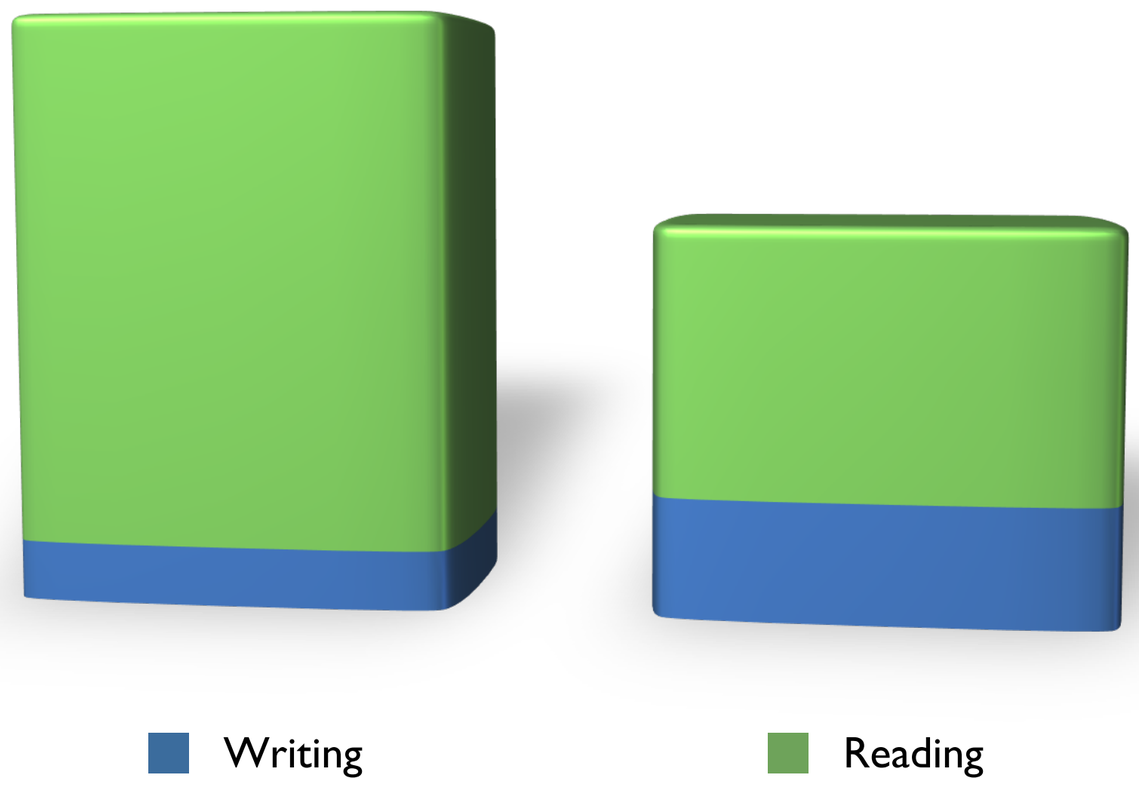
How much of our time do we spend on actually typing code?

10%

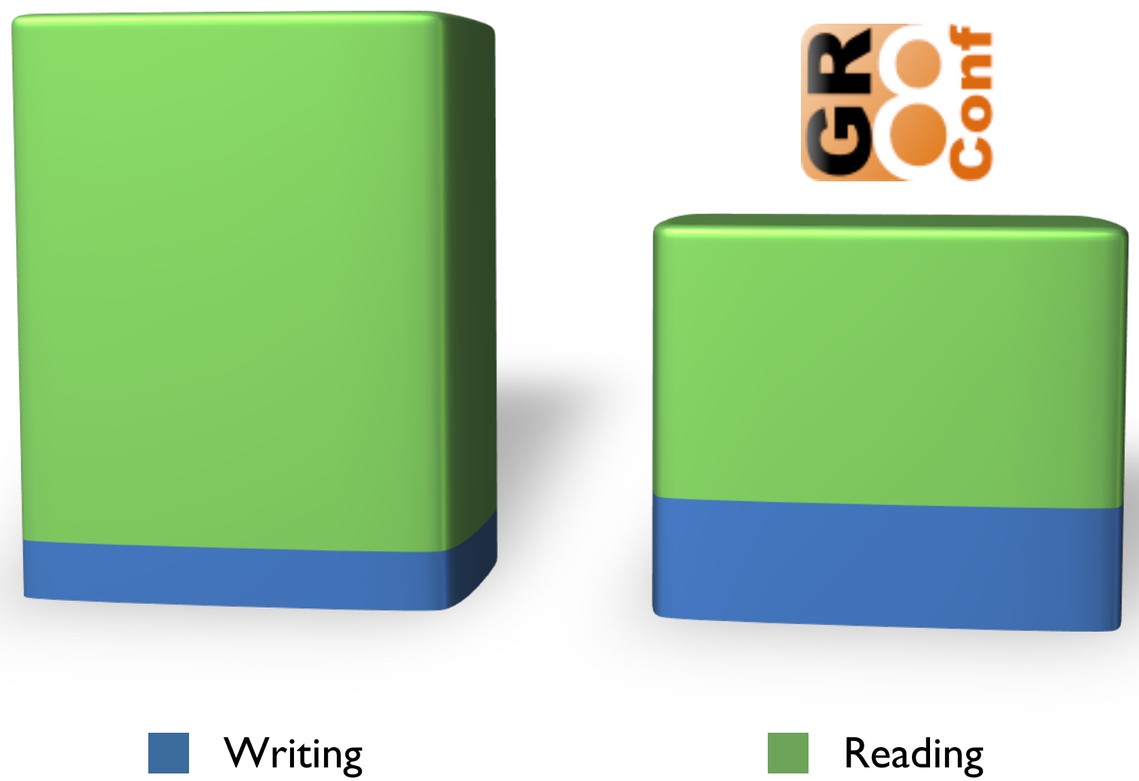
What if we could half the reading time by writing twice as much?



What if we could half the reading time by writing twice as much?

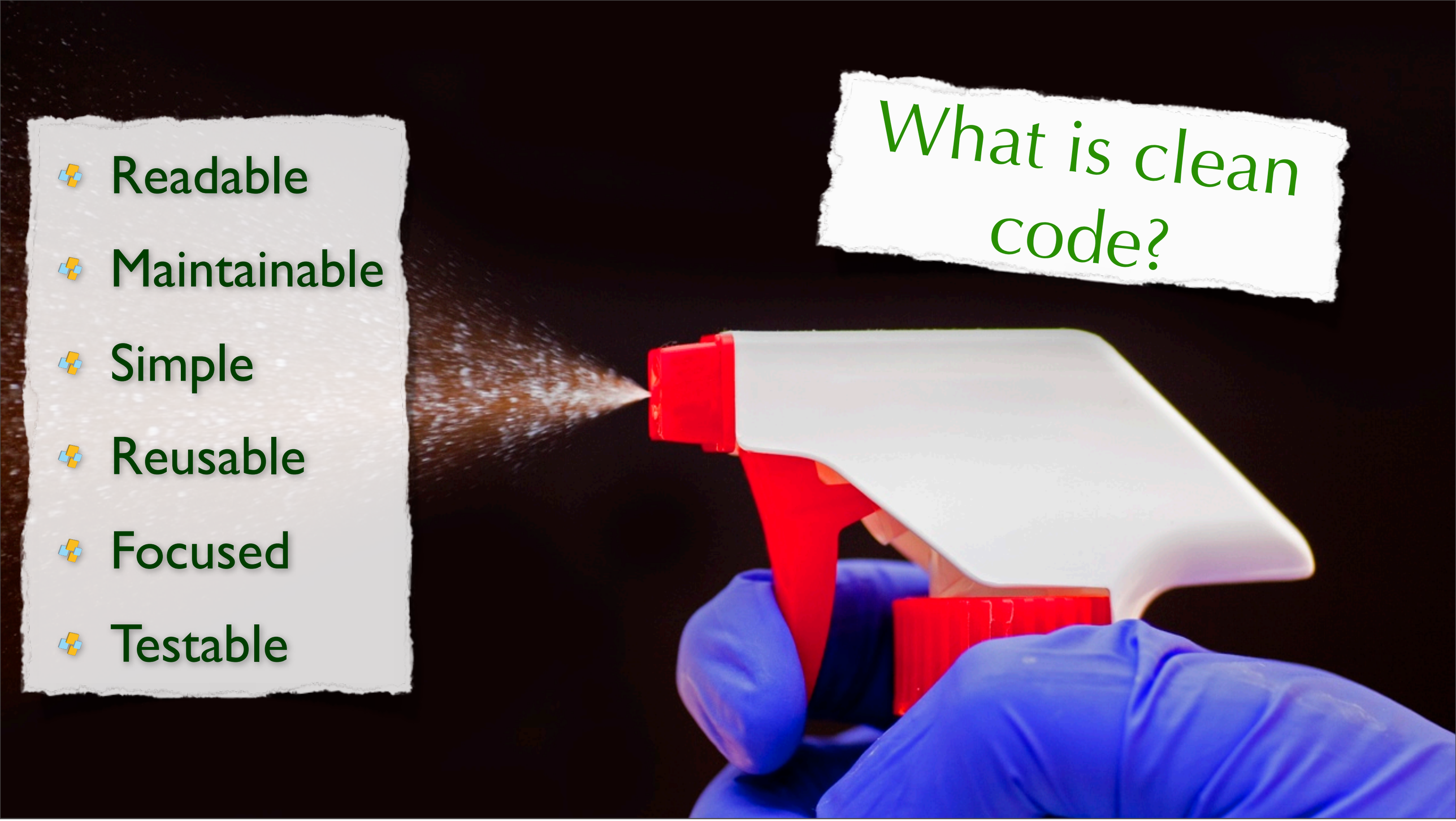


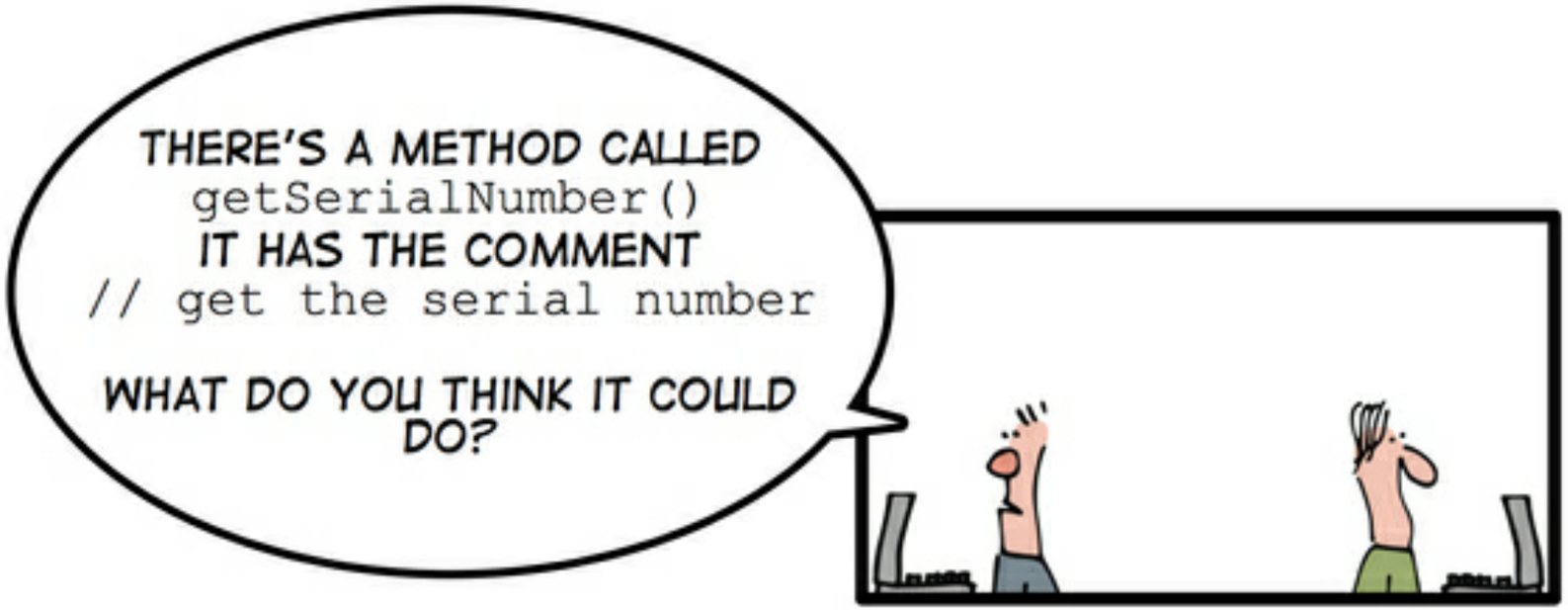
What if we could half the reading time by writing twice as much?



What is clean code?

- Readable
- Maintainable
- Simple
- Reusable
- Focused
- Testable

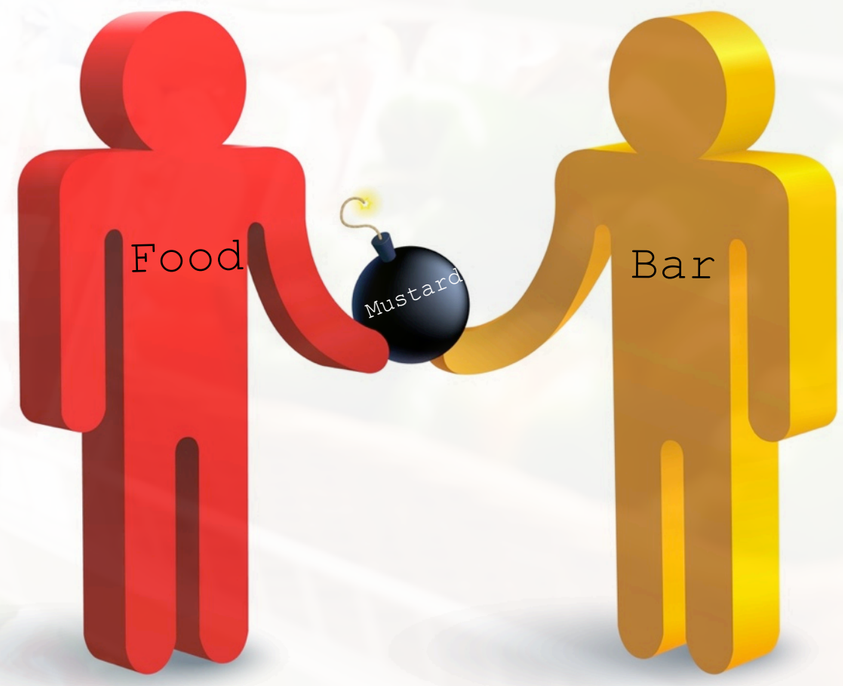



THERE'S A METHOD CALLED
getSerialNumber()
IT HAS THE COMMENT
// get the serial number
WHAT DO YOU THINK IT COULD DO?

Long garlic
Long egg
String mustard

```
Long garlic
Long egg
String mustard

if (mustard == "completed"){
    garlic ++
} else {
    egg ++
}
```



```
int y
```

```
int y
int years
```

```
int y
int years
int yearsOld
```

```
int y
int years
int yearsOld
int carAgeInYears
```

```
int y // how old the car is in years
int years
int yearsOld
int carAgeInYears
```

```
Integer calculate(List t) {
   Integer res = 0
   for (s in t) {
      if (!s.children.isEmpty()) {
         res += calculate(s.children)
      } else {
         res ++
      }
   }
   return res
}
```

Calculate something?

Set a local variable to 0.

Based on a list of t:s

```
Integer calculate(List t) {
  Integer res = 0
  for (s in t) {
    if (!s.children.isEmpty()) {
      res += calculate(s.children)
    } else {
      res ++
    }
  }
  return res
}
```

For every s in the list of t:s do the following

Check if the children of s is empty

If so, calculate them too

Otherwise, just add 1

Calculate something?

Set a local variable to 0.

Based on a list of t:s

```
Integer calculate(List t) {
    Integer res = 0
    for (s in t) {
        if (!s.children.isEmpty()) {
            res += calculate(s.children)
        } else {
            res ++
        }
    }
    return res
}
```

For every s in the list of t:s do the following

Check if the children of s is empty

If so, calculate them too

Otherwise, just add 1

```
Integer calculate(List t) {
   Integer res = 0
   for (s in t) {
      if (!s.children.isEmpty()) {
         res += calculate(s.children)
      } else {
         res ++
      }
   }
   return res
}
```

```
Integer countLeafStepsIn(List<TestStep> testSteps) {
   Integer leafCount = 0;
   for (TestStep testStep in testSteps) {
      if (testStep.isGroup()) {
         leafCount += countLeafStepsIn(testStep.children)
      } else {
         leafCount ++
      }
   }
   return leafCount
}
```

```
Integer calculate(List t) {
   Integer res = 0
   for (s in t) {
      if (!s.children.isEmpty()) {
         res += calculate(s.children)
      } else {
         res ++
      }
   }
   return res
}
```

Better method name

Clearer parameter name

Now we know what we're counting?

```
Integer countLeafStepsIn(List<TestStep> testSteps) {
   Integer leafCount = 0;
   for (TestStep testStep in testSteps) {
      if (testStep.isGroup()) {
         leafCount += countLeafStepsIn(testStep.children)
      } else {
         leafCount ++
      }
   }
   return leafCount
}
```

Method call instead of boolean expression

Methods

- One method should do only one thing, in only one abstraction level
- Methods should be short.
- The name should clearly reveal the methods intent.
- Methods should have a minimal number of arguments.

```
class ConferenceController {

    def attendeeEmailsByCountry(Long conferenceId) {

        Conference conference = Conference.get(conferenceId)
        List<Attendee> attendeesByCountry = conference?.attendees?.sort { it.address.country }

        /* Create a list of country maps with attendees emails
              [ name: "Sweden", attendees: [[ email: 'andreas.arledal@find-out.se' ] ] ]
        */
        List attendeeEmailsPerCountry = []
        String currentCountry
        attendeesByCountry.each { Attendee attendee ->
            if (attendee.address.country != currentCountry) {
                attendeeEmailsPerCountry << [name: currentCountry, attendees: []]
                currentCountry = attendee.address.country
            }
            attendeeEmailsPerCountry[-1].attendees << [ email: attendee.email ]
        }

        render view: 'emailsByCountry', model: [attendeeEmailsPerCountry: attendeeEmailsPerCountry]

    }

}
```

Get the attendees by country

Generate a structured list

Render the view and pass the model

```
class ConferenceController {
    def conferenceService

    def attendeeEmailsByCountry(Long id) {
        render view: 'emailsByCountry',
               model: [
                   attendeeEmailsPerCountry:
                   conferenceService.attendeeEmailsPerCountryForConference(id)]
    }
}

class ConferenceService {
    List attendeeEmailsPerCountry(Long conferenceId) {
        Conference conference = Conference.get(conferenceId)
        generateAttendeeEmailsByCountryFrom(conference?.attendeesByCountry)
    }

    private List generateAttendeeEmailsByCountryFrom(List<Attendee> attendeesByCountry) {
        List attendeeEmailsPerCountry = []
        String currentCountry

        attendeesByCountry.each { Attendee attendee ->
            if (attendee.country != currentCountry) {
                attendeeEmailsPerCountry << [name: currentCountry, attendees: []]
                currentCountry = attendee.address.country
            }
            attendeeEmailsPerCountry[-1].attendees << [ email: attendee.email ]
        }
        return attendeeEmailsPerCountry
    }
}
```

Render the view and pass the model

Get the attendees by country

Generate a structured list

Classes

- Classes should be small
- One class should have at most one reason to change
- Classes should have a small number of instance variables

```
class Radio {

   void changeStation() {
      // Implementation
   }
   void volumeUp() {
      // Implementation
   }
   void volumeDown() {
      // Implementation
   }

}
```

```
class StationManager {

   void change() {
      // Implementation
   }

}

class VolumeManager {
   void increase() {
      // Implementation
   }
   void decrease() {
      // Implementation
   }
}

class Radio {

   StationManager stationManager
   VolumeManager volumeManager

}
```

What defines clean code?

- Readable ✓
- Maintainable✓
- Simple ✓
- Reusable ✓
- Focused ✓
- Testable ?

The best way to write testable code is to write the tests first

Stop at the red light

Wait for green

Do a U-turn

Model

Handle business domain data and it's logic

- Domain specific logic should be placed in the domain class.
- Query your Queries with Where Queries

```
class Conference {

    static hasMany = [attendees: Attendee, speakers: Speaker]

    String name
    Date startDate
    Date endDate

     static constraints = {
        name blank: false
        startDate()
        endDate validator: { val, obj ->
            return val.after(obj.startDate)
        }
     }

    static namedQueries = {
        /* ... */
    }

    String toString() {
        "${name}"
    }

    Integer getNumberOfAttendeesAndSpeakers() {
        getNumberOfAttendees() + getNumberOfSpeakers()
    }

    private Integer getNumberOfAttendees() { /* ... */ }
    private Integer getNumberOfSpeakers() { /* ... */ }

}
```

Class variables*

Instance variables

Constraints

Named Queries

Public methods

Private methods

View

Present data to the user

- Don’t put business logic in the view.
- Use templates or taglibs for stuff that is repeated.
- Use templates to keep your views simpler.
- Layouts can be applied to layouts

Controller

It's the controllers responsibility to handle request flow

- The controller actions should do as little of the work as possible.
- Let your services do the work.
- If you need to map a request to something that is not a domain class or to a part of a domain class, use Command Objects.

```
class ConferenceController {

    def conferenceService

    def listUpcomingConferences() {

        List<Conference> upcomingConferences = Conference.upcoming.list()

        withFormat {
            html upcomingConferences: upcomingConferences
            json { render upcomingConferences as JSON }
            xml { render upcomingConferences as XML }
        }
    }

    def financeReportForConference(Long id) {

        Map financeReportData = conferenceService.generateReportDataForConference(id)

        withFormat {
            html financeReportData: financeReportData
            json { render financeReportData as JSON }
            xml { render financeReportData as XML }
        }

    }

}
```

Response directly from the domain class

Information needs processing.

Services

Services responsibility is to handle the application logic

- Never store state in a service.
- Services are singletons by default.
- You can inject other services into a service.
- Services are transactional by default.

Don’t forget about the src/groovy folder

- Not everything is a Grails artifact.
- If you need validation you can use @Validateable.
- When you are tempted to store state in a Service you should probably create a separate class instead.

Taglibs

Taglibs can be used to avoid coupling and to keep your views DRYer

- Create a tag if you reference something that might change.
- Use tags to avoid repetition, but keep in mind that they should be short, otherwise use a template.
- Make sure you use the Single Responsibility Principle in your taglibs.

```
<script type="text/javascript">
```

JavaScript

Javascript should be responsible for the behaviour of your views

- Don't use Grails tags for javascript except for including resources.
- Don't use JQuery for everything.
- Decide on a structure and stick to it.
- Learn about JavaScript patterns

Treat Javascript
as it's your best friend

CodeNarc Report

Report title:	Sample Report
Date:	2012-mar-26 08:42:28
Generated with:	CodeNarc v0.16.1

CODENARC
Less Bugs Better Code

Summary by Package

Package	Total Files	Files with Violations	Priority 1	Priority 2	Priority 3
All Packages	46	-	-	-	-
grails-app/controllers/com/maquet/provis	7	-	-	-	-
grails-app/domain/com/maquet/provis	13	-	-	-	-
grails-app/services/com/maquet/provis	3	-	-	-	-
src/groovy/com/maquet/provis	2	-	-	-	-
src/groovy/se/findout/diagram	17	-	-	-	-
src/groovy/se/findout/exceptions	3	-	-	-	-
test/unit/com/maquet/provis	1	-	-	-	-

JSHint

As a user **I would** like to be able to change my generated password **So that** I can have a password I remember

Links

- Clean Code
- Where Queries
- Applying Layouts in Layouts
- Javascript Design patterns

Thank you!

@AndreasArledal