Drei Arten Tests zu schreiben:

- xunit: loging mein code, klassen, einfach
- bdd: jquery plugin mein code, kontexte, verschachtelt, komplizierter
- cucumber: cucumber beispiel en, plain text, kommunikativ

Testen auf verschiedenen Levels (System Under Test SUT):

- methode
- klasse
- modul
- subsystem
- ganzes system

TDD, BDD

- notation, inside out vs. outside in
- für Kennt Beck das gleiche
- bdd: hat innovation im testing bereich angetrieben
- different styles of testing: london style (interaction) / classic (triangulation)

utilities:

- mocks: eigene frameworks
  - constraint matchers
  - how to use mocks
- matchers: bdd frameworks
- refactoring tools: IDEs, search/ replace

Testing bigger Systems:

- testing helpers: common setup operations -> might indicate high coupling
- page objects / ui maps: might indicate too low level cutoff point

Possible Goals of testing

- good rythm: red, green, refactor
- fast startup time after distraction / at morning
- reduce number of bugs per LOC
- document code / features
- regression controll / bug repellant
- know when done / automate acceptance criterias
- improve implementation speed
- improve / document code speed
- reduce debugging time / defect localization
- allow refactoring (deeper / more risky / faster / with less tool support)
- work with confidence: my changes don't break the system, even with limited understanding (junior!)
- continuus integraton
- continuus deployment
- always working nightly build
- improved scalability of project (add more people more easily)

How much test code:

- 50/50 common
- depends on the problem: compiler might be 20/80 or more

Common Problems:

- time to maintain tests
  - big tests
  - brittle tests
  - wrong level of abstraction
- slow to execute
  - wrong level of abstraction
  - too many acceptance tests, too little unit tests
- test code breaks application
  - test code in application causes problems
- not executed -> rotting tests
  - too complicated to do
  - no continuus integration server

Ideal Testsuite:

- fast
- complete
- easy to understand
- easy to execute

Often hard to test: -ilities / System Qualities

- usability
- dependability
- surviveability
- predictability
- maintainability
- reliability
- reuseability
- interoperability
- availability
- scalability
- flexibility
- understandability
- portability
- security
- safety
- manageability
- adaptability
- configurability
- supportability
- extensibility
- …