a manager’s story

story from:
“Brilliance”; in Software Requirements and Specifications:
A Lexicon of Practices, Principles and Prejudices;
Michael Jackson; Addison Wesley, 1995.
a manager’s story

Fred as Frankenstein
creator of the monster flow chart
not even sure he understands it himself!

story from:
“Brilliance”; in Software Requirements and Specifications:
A Lexicon of Practices, Principles and Prejudices;
Michael Jackson; Addison Wesley, 1995.
a manager’s story

Fred as Frankenstein
creator of the monster flow chart
not even sure he understands it himself!

Jane, a disappointment
gave her hard problems, but they turned out to be simple

story from:
“Brilliance”; in Software Requirements and Specifications:
A Lexicon of Practices, Principles and Prejudices;
Michael Jackson; Addison Wesley, 1995.
my favorite software quote
I gave desperate warnings against the obscurity, the complexity, and over-ambition of the new design, but my warnings went unheeded. I conclude that there are two ways of constructing a software design: One way is to make it so simple there are obviously no deficiencies and the other way is to make it so complicated that there are no obvious deficiencies.

Tony Hoare, Turing Award Lecture, 1980
some simple things?
some simple things?
some simple things?
some simple things?
some simple things?
some simple things?
some tactics
some tactics

focus on problem, not solution

magic alignment (Intellidraw)
some tactics

focus on problem, not solution
  magic alignment (Intellidraw)

question every assumption
no universal datatype? (WWW)
some tactics

focus on problem, not solution
magic alignment (Intellidraw)

question every assumption
no universal datatype? (WWW)

everything's a ...
file (Unix), formula (VisiCalc), object (Smalltalk)
butler lampson on the WWW
butler lampson on the WWW

My view about the web is that it’s the great failure of computer systems research. Why did computer systems researchers not invent the web? And I can tell you the answer. It’s because it’s too simple...
My view about the web is that it’s the great failure of computer systems research. Why did computer systems researchers not invent the web? And I can tell you the answer. It’s because it’s too simple...

If I had been there I would have mucked it up. I swear to God. The idea that you’re going to make a new TCP connection for every mouse click on a link? Madness! The idea that you’re going to have this crusty universal data type called HTML with all those stupid angle brackets? We never would have done that! But those were the things that allowed it to succeed.

Butler Lampson
interviewed by Alan Kay in 2006 for the Computer Museum
use the good, shun the bad
use the good, shun the bad

It is rarely possible for standards committees to remove imperfections from a language... But you have the power to define your own subset.

Douglas Crockford, in *Javascript: The Good Parts*
use the good, shun the bad

It is rarely possible for standards committees to remove imperfections from a language... But you have the power to define your own subset.

Douglas Crockford, in *Javascript: The Good Parts*

*The Skater’s Principle of Language Use:*
Stay in the middle, where the ice is thicker.

*Michael Jackson, in Software Requirements & Specifications*
use the good, shun the bad

It is rarely possible for standards committees to remove imperfections from a language... But you have the power to define your own subset.

Douglas Crockford, in Javascript: The Good Parts

The Skater’s Principle of Language Use:
Stay in the middle, where the ice is thicker.

Michael Jackson, in Software Requirements & Specifications

There is danger and misery at the edges.

Douglas Crockford, in Javascript: The Good Parts
equals in JS
equals in JS

> 0 == ''
true
> 0 == '0'
true
> '' == '0'
false
equals in JS

> 0 == ''
true

> 0 == '0'
true

> '' == '0'
false

> 0 === ''
false

> 0 === '0'
false

> '' === '0'
false
equals in JS

> 0 == ''  > 0 === ''
  true       false
> 0 == '0'  > 0 === '0'
  true       false
> '' == '0'
  false
> '' === '0'
  false

**lesson**
always use ===, not ==
a relentless pursuit of simplicity
simplicity comes slowly...
simplicity comes slowly...

Inside every large program there is a small program trying to get out.

Tony Hoare
simplicity comes slowly...

Inside every large program there is a small program trying to get out.

Tony Hoare

Simplicity does not precede complexity, but follows it.

Alan Perlis
a coding story
don’t debug
don’t debug

instead, get it right the first time

cost of mistakes x10 for each stage

start on paper: forces you to think

WISCY: “why isn’t Sam coding yet?”

know when perfection is needed
if you must debug...
if you must debug...

be Sherlock Holmes
articulate every assumption
examine evidence: what changed?
devise experiments
if you must debug...

be Sherlock Holmes
articulate every assumption
examine evidence: what changed?
device experiments

if no progress after 10 minutes
explain it to someone else
take a walk
what’s really going on?
what’s really going on?

```html
<script>
function click () {
    alert("clicked!");
}
</script>
<input id="x" onclick="click()"></input>

fails
what’s really going on?

fails

```html
<script>
function click () {
    alert("clicked!");
}
</script>
<input id="x" onclick="click()"></input>
```

works

```html
<script>
function myClick () {
    alert("clicked!");
}
</script>
<input id="x" onclick="myClick()"></input>
```
what’s really going on?

lesson

`click` is a DOM method for element
real solution: unobtrusive Javascript
unison file sync
unison file sync

[chim:-] dnj% sync -path Teaching
2011-08-03 10:59:33.745 Unison[13422:f07] Call
Contacting server...
Password:
Connected [ //chim//Users/dnj/Filestore -> //cs
Looking for changes
   Waiting for changes from server
Reconciling changes

  csail    local
  <----- new dir  Teaching/Fall 11  (61
  <----- new file  Teaching/Fall 11  (61
  <----- changed  Teaching/Fall 11  (61
  <----- new file  Teaching/Fall 11  (61
  <----- new file  Teaching/Fall 11  (61
  <----- new file  Teaching/Fall 11  (61
  <----- new file  Teaching/Fall 11  (61

Proceed with propagating updates? [] y
Propagating updates
Propagation of Changes

The first property a correct synchronizer must satisfy is that it must never overwrite a change made by the user. This requirement can be formalized as follows:

A run \((o, a, b, a', b')\) is said to preserve user changes (locally) if

\[
\neg(o \sim a) \implies a' \sim a \\
\neg(o \sim b) \implies b' \sim b.
\]
Propagation of Changes

The first property a correct synchronizer must satisfy is that it must never overwrite a change made by the user. This requirement can be formalized as follows:

A run \((o, a, b, a', b')\) is said to preserve user changes (locally) if
\[
\neg (o \sim a) \implies a' \sim a \\
\neg (o \sim b) \implies b' \sim b.
\]

lesson
take the trouble to understand deeply
the beginning of wisdom
The beginning of wisdom for a programmer is to recognize the difference between getting his program to work and getting it right. A program which does not work is undoubtedly wrong; but a program which does work is not necessarily right. It may still be wrong because it is hard to understand; or because it is hard to maintain as the problem requirements change; or because its structure is different from the structure of the problem; or because we cannot be sure that it does indeed work.

first browser war: IE3 vs Navigator

Michael Cusumano and David Yoffee
Competing on Internet Time: Lessons from Netscape and Its Battle with Microsoft
Free Press, New York, 1998
first browser war: IE3 vs Navigator

Michael Cusumano and David Yoffee
Competing on Internet Time: Lessons from Netscape and Its Battle with Microsoft
Free Press, New York, 1998
first browser war: IE3 vs Navigator

Michael Cusumano and David Yoffee

Competing on Internet Time: Lessons from Netscape and Its Battle with Microsoft

Free Press, New York, 1998
first browser war: IE3 vs Navigator

netscape market share

lesson

Michael Cusumano and David Yoffe
Competing on Internet Time: Lessons from Netscape and Its Battle with Microsoft
Free Press, New York, 1998
first browser war: IE3 vs Navigator

netscape market share

lesson
take the time to restructure

Michael Cusumano and David Yoffee
Competing on Internet Time: Lessons from Netscape and Its Battle with Microsoft
Free Press, New York, 1998
growing a program
growing a program

be incremental
build confidence as you go
break one thing at a time
take the time to polish
write comments, for yourself!
growing a program

be incremental
build confidence as you go
break one thing at a time
take the time to polish
write comments, for yourself!
be like a beaver!

small nibbles, big outcome
on "optimization"
on “optimization”

...Premature optimization is the root of all evil... A good programmer ... will be wise to look carefully at the critical code, but only after that code has been identified.

— Donald Knuth, in ‘Structured Programming with Goto Statements’
on “optimization”

...Premature optimization is the root of all evil... A good programmer ... will be wise to look carefully at the critical code, but only after that code has been identified.

— Donald Knuth, in ‘Structured Programming with Goto Statements’

Rules for Optimization:
Rule 1: Don’t do it
Rule 2 (for experts only): Don’t do it yet

...Premature optimization is the root of all evil... A good programmer ... will be wise to look carefully at the critical code, but only after that code has been identified.

— Donald Knuth, in ‘Structured Programming with Goto Statements’

*Rules for Optimization:*

Rule 1: Don’t do it
Rule 2 (for experts only): Don’t do it yet

self-reflection
self-reflection

how to become great
learn from your mistakes
repeat your successes