

# The Psychology of Developer Tool Usability

Slides + Notes + More at  
**<https://git.io/DevTools>**

@saraforde - Microsoft  
3<sup>rd</sup> time's a charm! -\\_(\`ツ)\\_/\\_



Thanks to our sponsors!

# The Psychology of Developer Tool Usability

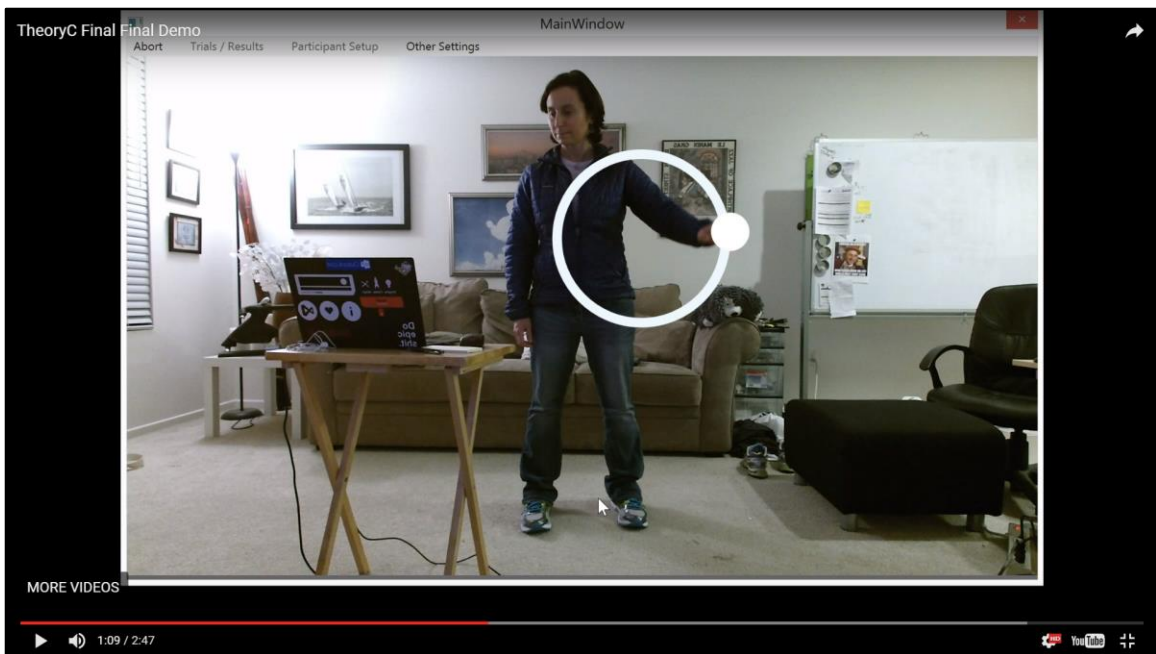
@sarafor

- **Many of you know me from CodePlex and Visual Studio tip of the day series**
- And some of you may know that I loved speaking internationally about CodePlex and Visual Studio

# The Psychology of Developer Tool Usability

@sarafor

- But what you don't know is
- in 2011, just shortly after moving to CA to train w my grand master in karate, why everything came to a screeching halt
- I was given a compression sock that was too tight and did severe nerve damage to the nerves in the skin in my left leg
- The damage got so bad I could barely walk or do a single rotation on a stationary bike
- **People would ask: did you get hurt from karate? I'm no. A sock. WHO GETS HURT FROM A SOCK??**
- PT 3 times a week for 1<sup>st</sup> year. 2 times a week 2<sup>nd</sup> year, 1 time 3<sup>rd</sup> year and so on and so forth until I got back into karate
- My leg doctor who is ... at Stanford came to watch me do karate one night w grand master to show his support
- And it was hands down the nicest thing anyone has ever done for me
- And as a thank you, I promised him I'd get my life back, I'd move heaven and earth to get back to doing the things I love, like traveling and public speaking and getting back on stage again
- And today is my 3<sup>rd</sup> time speaking at a conference since then!



What does a person who can never sit still do during that much “downtime”? Get a Master’s degree!

Masters - make dev tools usable

Built a Kinect app

1960s Rotary pursuit

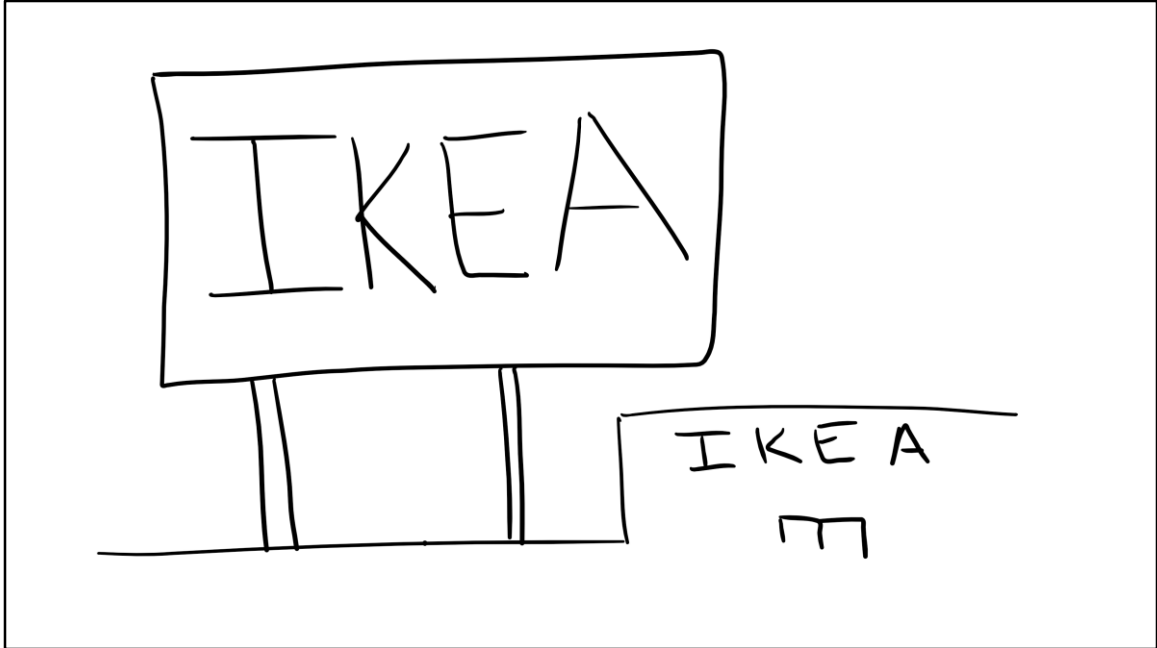
And after a year of building this app, when the researcher went to use it, she struggled immediately with the usability of the app.

I’ve just spent 3.5 years getting my grad degree and my app is unusable!!!

**Then I had a lightbulb moment that you’ll hear about in a second, but first some context.**

# Prescriptive vs Descriptive

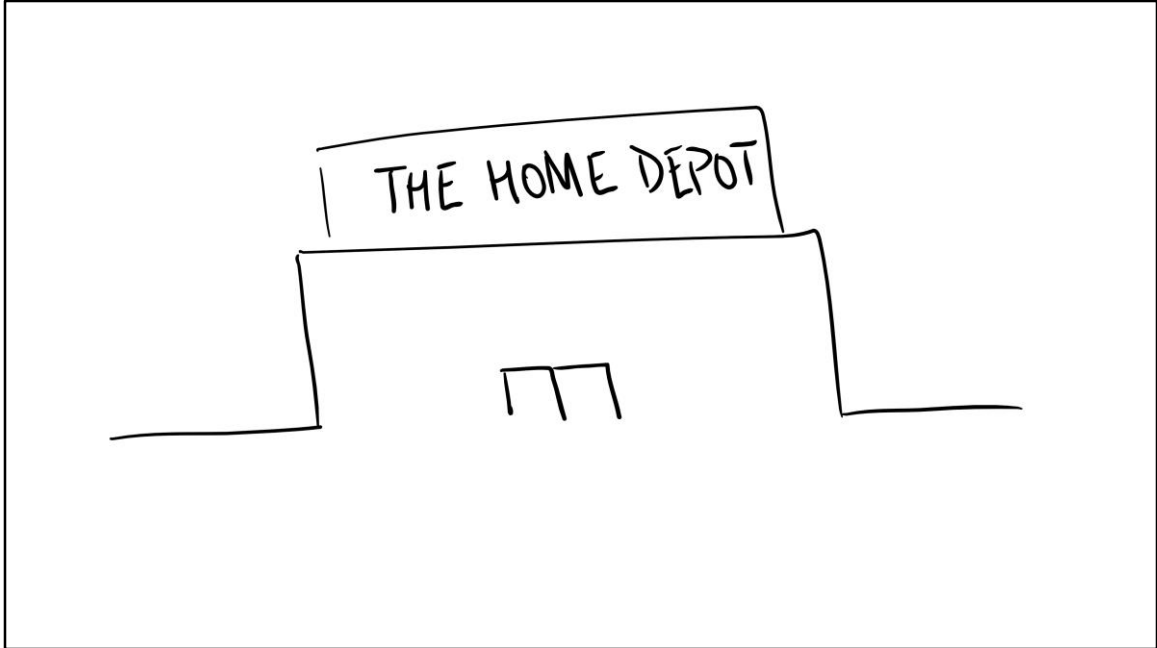
Instead of quality vs quantity, let's think in terms of Prescriptive vs Descriptive  
<NEXT SLIDE>



**Prescriptive** – you're told exactly how to do something every single time  
There's the top down - traditional approach.

- Go collect requirements
- Code written to requirements

**Like Ikea furniture**



**descriptive** – you're told how things are done in general  
you get much more flexibility but you'll need to do more work

collect tools and lumber  
Same trade-off in dev tools

**bottom up design –**

- what is technologically possible, and so you build it.
- e.g. in my masters project – here are 50 possible configurations!
- **Thank you databinding**

\*\*\*\*\***It's more COMPLEX**\*\*\*\*\*, and often not the most usable, but it will get the job done.

<NEXT SLIDE>



**Just because something is complex  
doesn't mean it can't be usable**

**- My Graduate Advisor**

#### **WATER BREAK**

This is your take home message.

If you leave with nothing else today, walk away with this.

**Just become something is complex doesn't mean it can't be usable.**

My graduate advisor and kinesiology professor bashed this into our heads over and over

Think about

- Cockpit Management Systems
- ride sharing apps and all the other apps we have that are smaller than an index card!

Complex things **can be made usable**.

- anseisology for neruosurgery

**<NEXT SLIDE>**

# Simplicity => Intelligence

My grad advisor stressed **the importance of being concise**

- We once had to read 10-20 research papers but present in 5 minutes

Complexity DOES NOT IMPLY intelligence.

- It's just the opposite.
- Simplicity implies intelligence.

This is why it is so hard to delete code.

- We feel it doesn't show how hard we worked when it gets deleted.
- I'll be the first to admit I'll leave it commented out before I'll delete it, even if I have it committed in Git. Hey baby steps!

**Simplicity implies the hard work. We just have to allow ourselves to believe it.**

<NEXT SLIDE>

## Software Development and Tool Usability

Brian Dillon  
Naval Surface Warfare Center, Dahlgren Division  
Virginia Tech  
Blacksburg, VA, USA  
bwdillon@vt.edu

Richard Thompson  
Naval Surface Warfare Center, Dahlgren Division  
Dahlgren, VA USA  
richard.p.thompson@navy.mil

**Abstract**—Tools are used at every stage of the software life cycle with particular recent emphasis on the maintenance period. Evidence shows that maintenance tools are underused, even by the developers who create them. Integrated development environments were created to empower developers, but they have remained virtually unchanged since the late 1980s. This paper examines the challenges of creating development tools, analyzes the usability of two frequently used tools, and suggests that poor tool usability may be inhibiting more efficient software development.

Developers' tools are designed from the bottom up, a term Panko and Bauer [8] used to indicate that technology drives the requirements. The approach might be phrased as "We can do X, therefore we will give the user the option to do X." Conversely, a top-down approach is built from and adapts to meet user requirements. Panko and Bauer also noted that a user is more likely to adapt to the system if it is constructed as bottom-up. Manual editing by developers, even when a tool exists, may be a form of adaptation.

## Brian Dillon and Richard Thompson

IEEE 24th International Conference  
on Program Comprehension 2016

**Now if you're still having a hard time believing me re dev tools...**

Here's an example I read about recently in the research literature.

- *Yes of course I read research articles on dev tool usability during vacation*

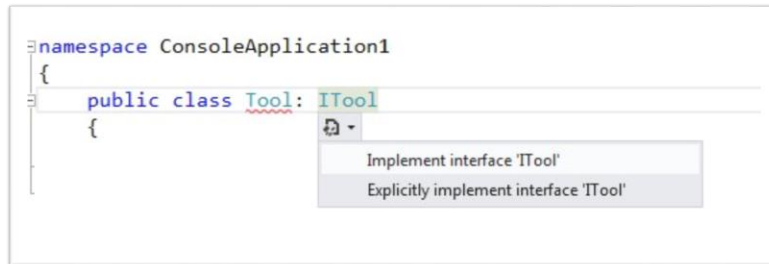
Examined challenges of IDEs  
Conducted usability studies  
Industry standard UX principles  
For tasks

- Add new method
- Renaming

VS 2012 and Eclipse

Here's an example of their findings

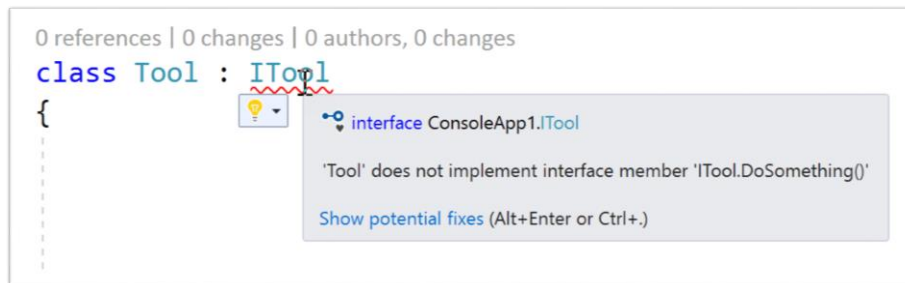
<NEXT SLIDE>



## Visual Studio 2012

Dillon and Thompson 2016

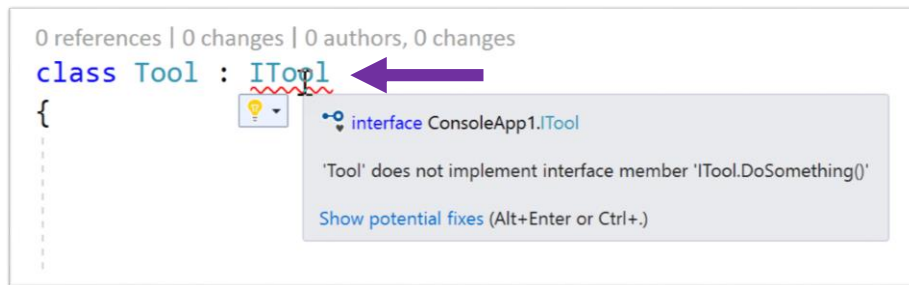
- In this example, a new method was added to the ITool interface.
- They suggested several UX improvements
  1. squiggle
  2. Transparency - I've already implemented ITool?
  3. And what to do next – "it's like the C# compiler is saying "you know what you did wrong"



## Visual Studio 2017

Dillon and Thompson 2016

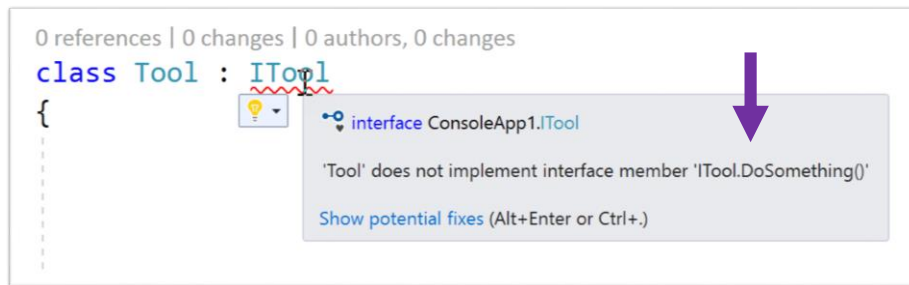
- VS 2017 happens to have addressed these 3



## Visual Studio 2017

Dillon and Thompson 2016

- VS 2017 happens to have addressed these 3



## Visual Studio 2017

Dillon and Thompson 2016

- VS 2017 happens to have addressed these 3

0 references | 0 changes | 0 authors, 0 changes

```
class Tool : ITool  
{  
    ...  
}
```



interface ConsoleApp1.ITool

'Tool' does not implement interface member 'ITool.DoSomething()'

Show potential fixes (Alt+Enter or Ctrl+.)



## Visual Studio 2017

Dillon and Thompson 2016

**Segue: if they can be made usable, why are we so far behind?**



## Human-Centric Development of Software Tools

Edited by  
Andrew J. Ko<sup>1</sup>, Shriram Krishnamurthi<sup>2</sup>, Gail Murphy<sup>3</sup>, and  
Janet Siegmund<sup>4</sup>

<sup>1</sup> University of Washington, USA, [ajko@cs.washington.edu](mailto:ajko@cs.washington.edu)

<sup>2</sup> Brown University, USA, [skrishna@brown.edu](mailto:skrishna@brown.edu)

<sup>3</sup> University of British Columbia, Canada, [murphy@cs.ubc.ca](mailto:murphy@cs.ubc.ca)

<sup>4</sup> University of Passau, Germany, [Janet.Siegmund@uni-passau.de](mailto:Janet.Siegmund@uni-passau.de)

### Abstract

Over two and half days, over 20 participants engaged in inventing and evaluating programming and software engineering tools from a human rather than tool perspective. We discussed methods, theories, recruitment, research questions, and community issues such as methods training and reviewing. This report is a summary of the key insights generated in the workshop.

Andrew J. Ko,  
Shriram Krishnamurthi,  
Gail Murphy, and  
Janet Siegmund

Report from Dagstuhl Seminar 15222



One takeaway

*“the industry standard UX principles often don’t consider the **culture and context** of software engineering.”*

30 UX researchers for Dev Tools got together to talk shop, like we’re doing here.

*“But HCI often doesn’t consider the culture and context of software engineering, and doesn’t address the longitudinal / long term factors in education and skill acquisition, and so HCI may not be a sufficient lens through which to understand software engineering.” page 117 (or page 3 if you just print out their paper)*

## Software Developers as Users: Developer Experience of a Cross-Platform Integrated Development Environment

Kati Kussinen<sup>(ID)</sup>

Tampere University of Technology, Korkeakoulunkatu 1, Tampere, Finland  
kati.kussinen@tut.fi

**Abstract.** Software development is professional activity that demands a plethora of skills and qualities from the developer. For instance, developers need technical skills to create the code that implements the running software and social skills to be able to collaborate with peer developers and with various stakeholders. Development is an endeavor towards building complex systems that realize user and business requirements in technologically sophisticated

**Kati Kussinen**

Product-Focused Software  
Process Improvement 2016

Key takeaway from this paper:

*It is unclear how these UX research methods apply to the software development (or bottom up approach)."*

*In other words, do UX principles apply **in the context** of developer tools?*

This paper conducted a **qualitative** (or exploratory) research study

Sent out 2 open-ended survey questions to developers globally

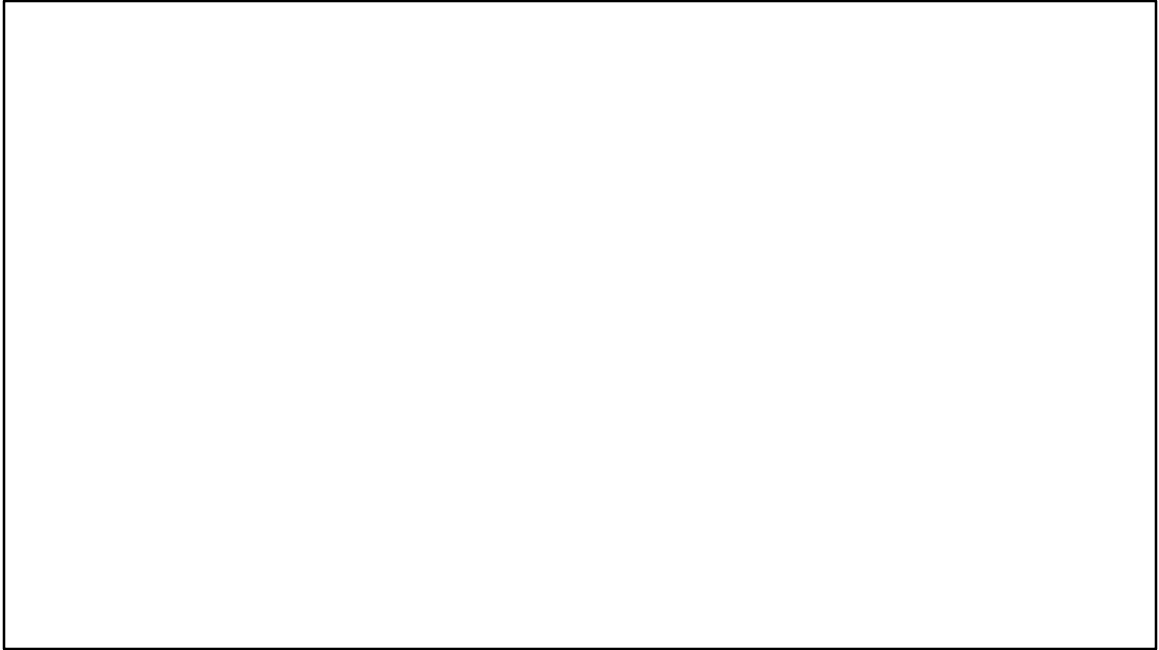
1. What are the best qualities of an IDE?
2. How could your IDE better support your development work?

In the paper, they contemplated

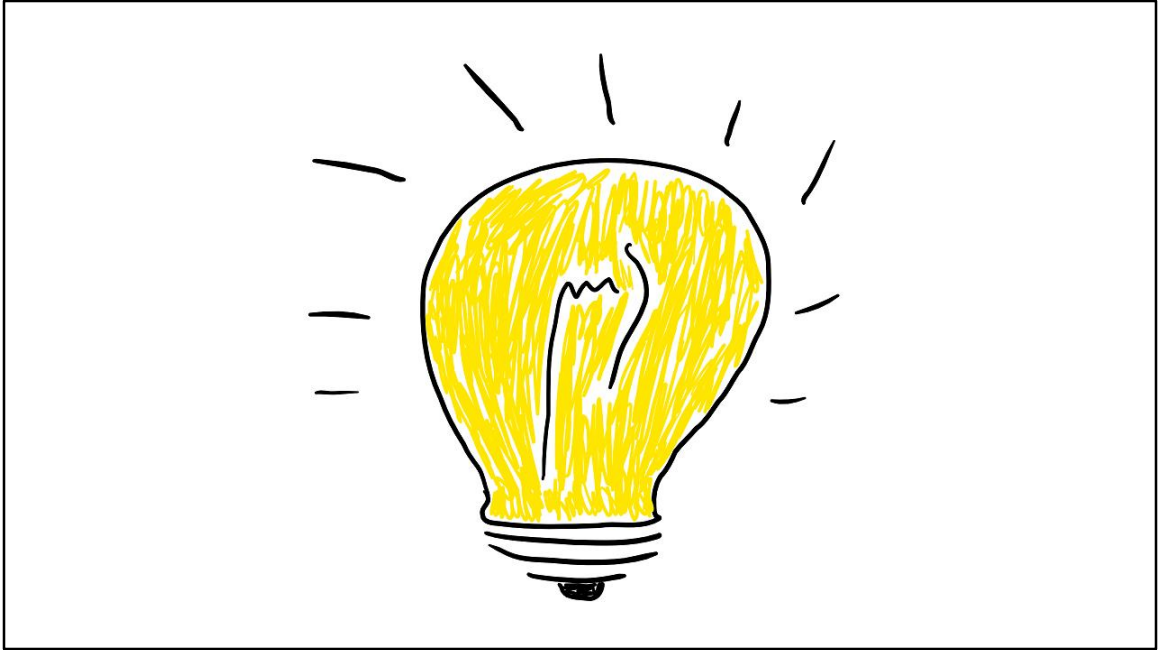
- "since UX studies concentrate mostly on the usability of end-user tasks...

*"Although early HCI studies concentrated almost exclusively on task- and work-related usability issues and achievement of behavioral goals [8], UX research has mainly concentrated on consumers and leisure systems [3]. Especially, studies in the software development context are lacking. Thus, it is unclear how these models and measurement scales serve the research of developer experience. Especially, studies in the software development context are lacking. Thus, it is unclear how these models*

*and measurement scales serve the research of developer experience” page 547 (or page 2 if printed)*



- So there I was... sitting in the lab - “how did I get this UX so wrong?”
- If I wasn’t focused on making the end user task more usable, what on earth was I so laser-focused on making usable over the course of an entire year.
- And that’s when it hit me!



- **It's all about context.**
- That's what all those research papers were alluding to.
- We're in a different context when building bottom-up, descriptive, home depot dev tools
- Which brings me to that lightbulb moment
- **Introducing.... @saraford's Theory of Developer Tool Unusability**

# **We design dev tools that are ideal for our own testing scenarios**

**- saraford theory of developer tool un-usability**

“We design for our own testing purposes”

- We think "ideal way to use" but it is “ideal way for us to test it”
- Too lazy to Wax-on / wax-off
- Minimizing the amount of keyboard typing and mouse-movements I needed to test.

**That’s my claim. Now for some psychology 101**



The moon illusion - As mentioned in abstract

- If you only take away one thing, let it be this...
- You know how you see the moon bigger on the horizon than you do looking straight up? It's an optical illusion. Yep, I swear. I had no idea. And unlike other illusions, you can't convince your eyes what they are seeing is wrong.
- No one knows why this is. Feel free to google "Optical Illusion" or read this article. <https://www.lhup.edu/~dsimanek/3d/moonillu.htm>

**Can anyone guess as a taste illusion?**

Hot sauce – it's an illusion of heat – the neurotransmitters just happen to match those for hot temperature detection

# **Sensation is objective**

# **Perception is subjective**

Sensation is

- it is the physics of information being sent from objects to nerves to brain
- It's **an engineering process**

Perception is

- the active mental process of interpreting sensory information



# **Sensation is deterministic Perception is non-deterministic**

In moon illusion, the sensation is correct (we're all processing the visual inputs correctly)

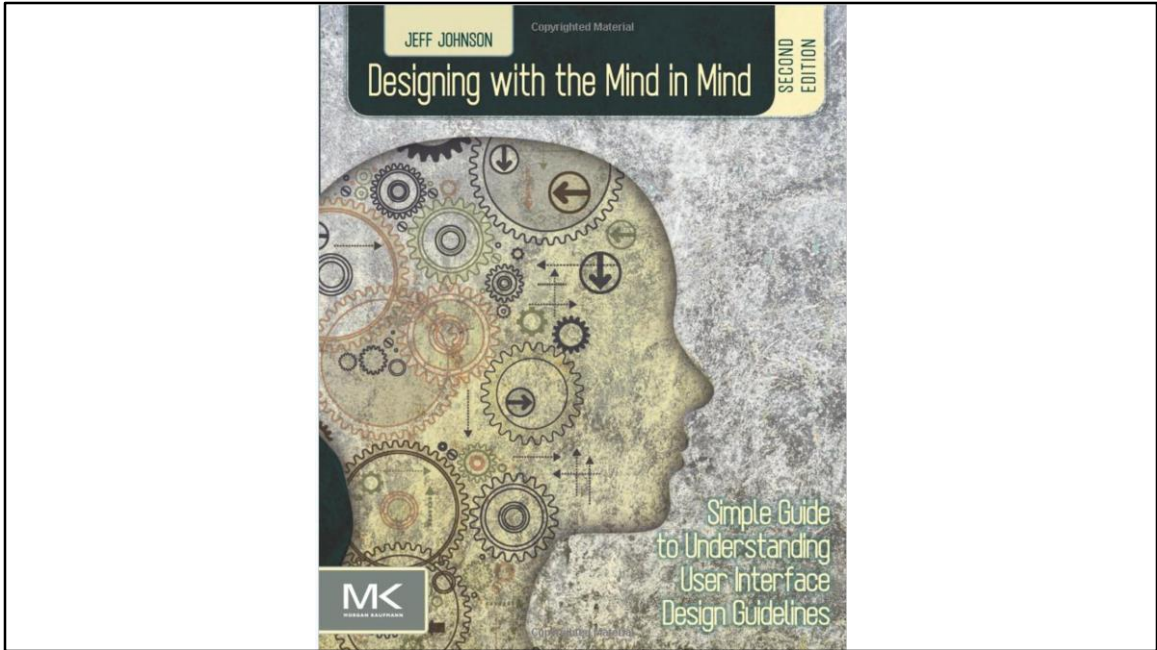
but the perception is off - the moon is same size. I'm interpreting it differently.

-----

If a tree falls in the forest, no it doesn't make a sound, because no one is there to perceive it.

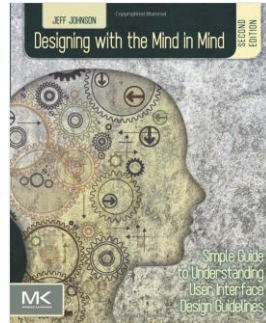
# Color does not exist

- This blew my mind. Turn it off, shut it down. Let's all go home.
- Color is a wavelength, an engineering process.
- Another way to look at it
  - **there's no way to compare our subjective experiences**, e.g. what I call blue might be what you call red.
- And this scares me half to death. If I'm holding this purple flyer, WHAT COLOR IS IT REALLY???
- What color is it?? If color doesn't exist, does this object exist? Do I exist?
- The unhandled exception just goes all the way up

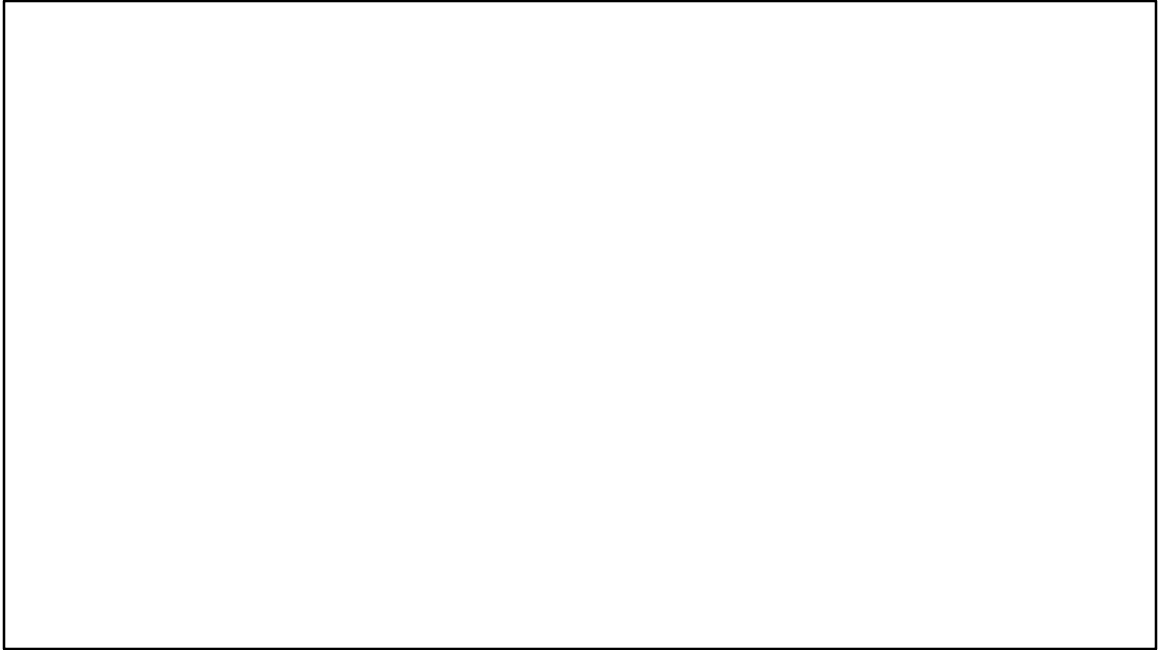


As an engineer, if you are interested in HCI or UX, this is the book to get  
<water break to let them take a photo>

# What we perceive is based on our experience



The very first page talks about how "What is perceived is based on our experience"  
And I'll prove this to you in a group exercise.



I have to read a quote directly from the book,

**“Imagine that you own a large insurance company. You are meeting with a real estate manager, discussing plans for a new campus of company buildings. The campus consists of a row of five buildings, the last two with T-shaped courtyards providing light for the cafeteria and fitness center. Now the real estate manager shows you the map...”**



10:34

<WATER BREAK>



If people aren't paying attention and they look up from their phones, they see the word life immediately and wonder what is going on

This sometimes confounds software designers, **who expect users to see what is on the screen**

- Designing with the Mind in Mind p4

This is my favorite quote.

Perception is based on prior experience.

**"This sometimes confounds software designers, who expect users to see what is on the screen."** - page 4

"Wait a sec, I gave you a building map, but you're seeing the word LIFE?!"



# Why testing scenarios?

If context is everything, why am I hypothesizing that we're designing for our own TESTING scenarios and not some other scenario?

# Explicit vs Implicit

## 2 types of knowledge

Explicit == declarative – knowledge you can verbalize

Implicit == procedural – knowledge you can't. it's unconsciousness. And it's usually procedural how to do something after we've mastered it.

## **Demo: Volunteer needed!**

Need a volunteer. Place your hands on your lap. Answer immediately. Did you ride a bicycle here today? Yes or No? Do you tie your shoestrings left over right or right over left? Answer immediately.

- But you know how to tie shoes strings – that procedure is implicit knowledge.
- That's why I told you to keep your hands on your lap. Your unconscious knows, even if you can't articulate it.

# Testing is implicit knowledge

- We've mastered "I need to verify what I just wrote"
- We have a strong desire to test code
- TDD (we write tests BEFORE we even write our code)



## Wendy Wood

Psychology Researcher in Habit  
Univ of Southern California  
<https://dornsife.usc.edu/wendywood>

### Testing is a matter of habit.

Researchers describe a habit as a repetitive behavior triggered by contextual cues in the environment that don't require conscious planning.

- You pull out your phone to check the time, and next thing you know, you're reading Twitter
- Habits can be desired behaviors (tying our shoes) or undesired (forgetting to put socks in laundry bin)
- Most everyday actions are habitual actions
- free up cognitive resources aka "reduces cognitive load"

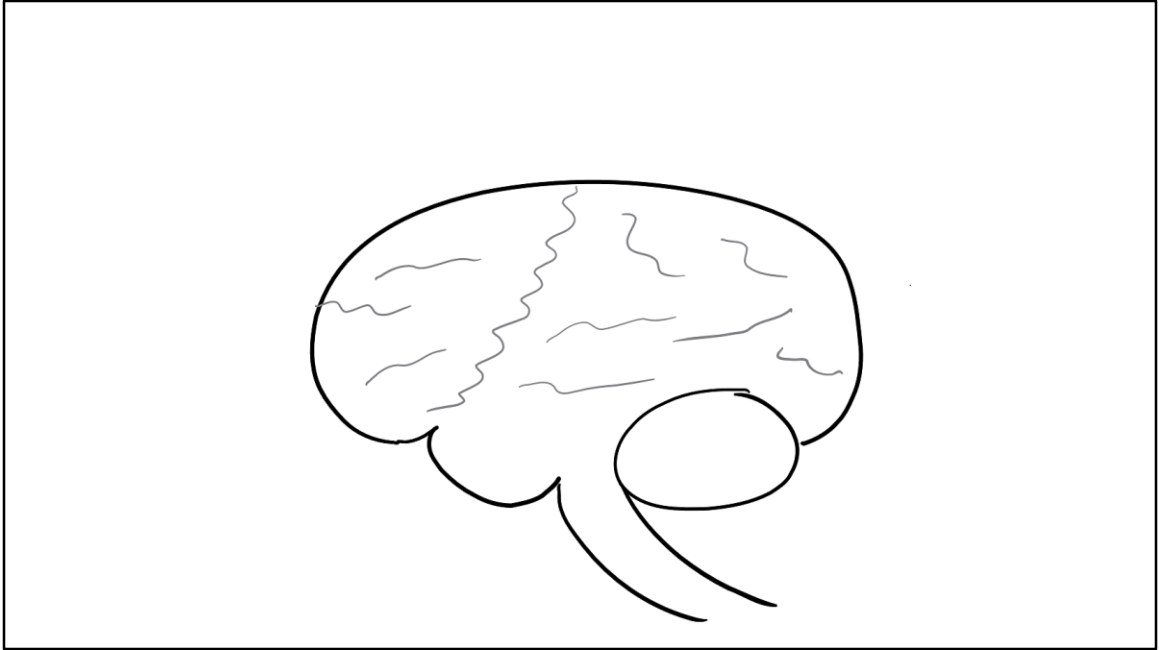
### Example of Cognitive load

- I am so visual I cannot listen to podcasts while driving. I can't visualize the podcast and pay attention to the road.

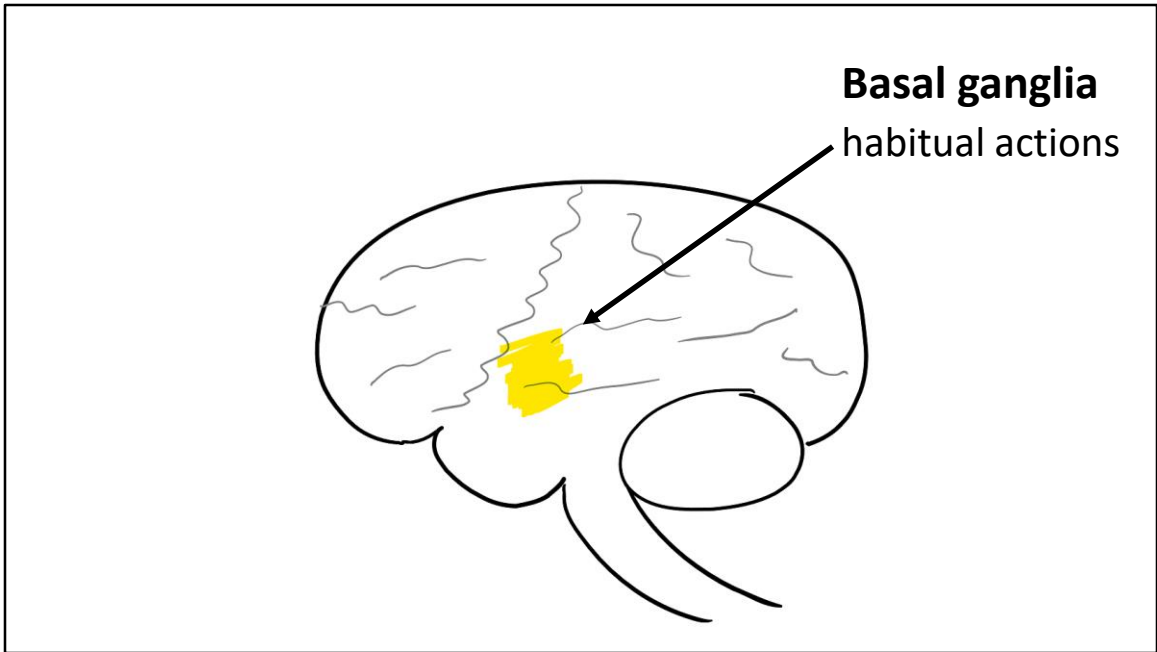
# How to break habit

- trigger is chunked as implicit procedural knowledge
- Don't know trigger b/c you can't articulate.
- must change upstream environmental cues.
- This is why new years resolutions fail
- Different than temptations which are goal-based
- Must change environment to break

**Cognition time!**

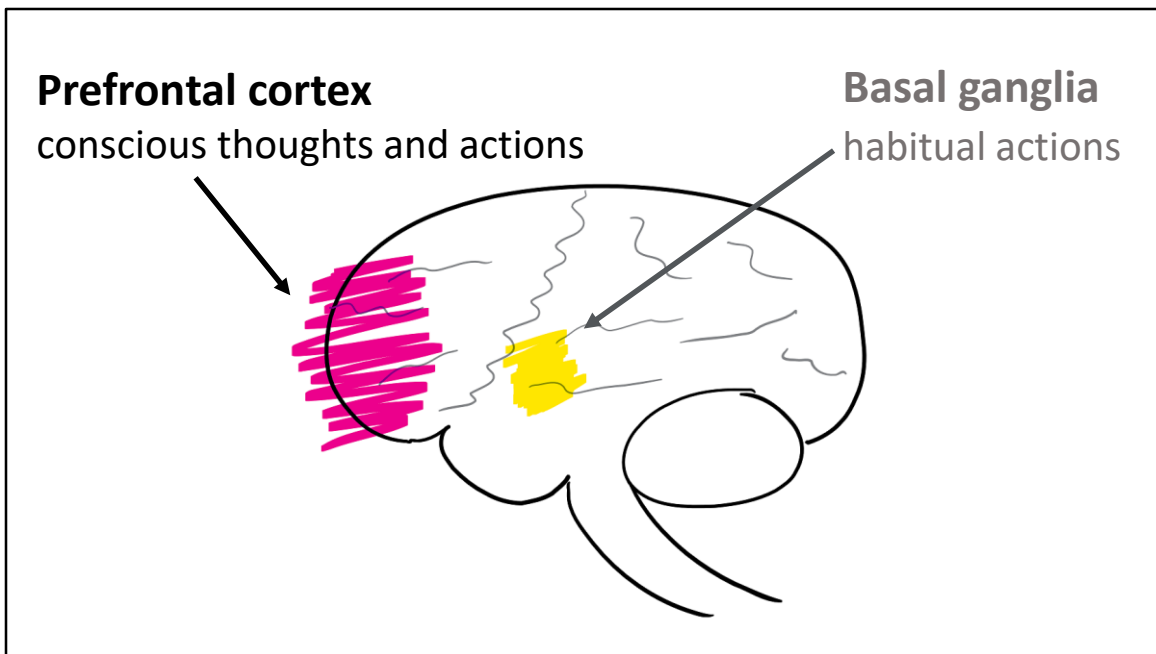


Yes, I drew that. I consider myself an abstract artist!



I know drawing is like Desktop = CPU  
“lizard brain” like the brain stem



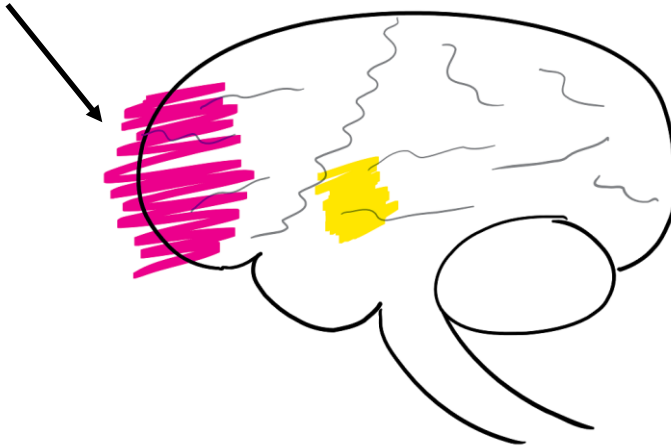


Frees up cognitive resources – reduces that cognitive load  
Although Kinesiology, still applies to coding – using hands for our craft

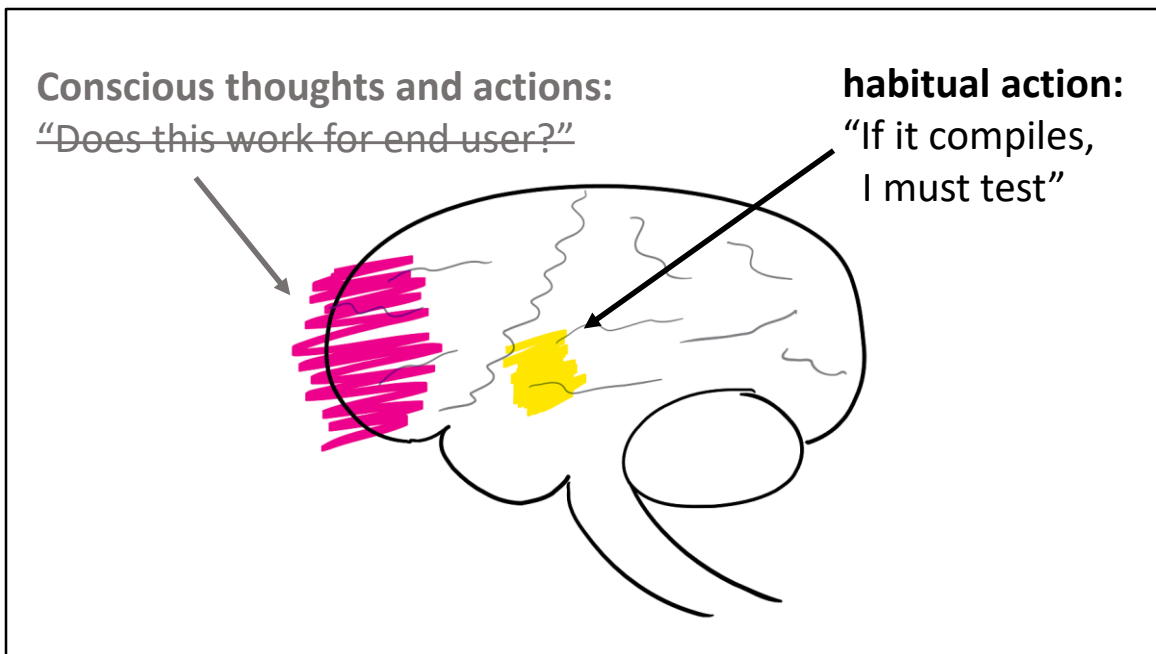
**Conscious thoughts and actions:**

“Does this work for end user?”

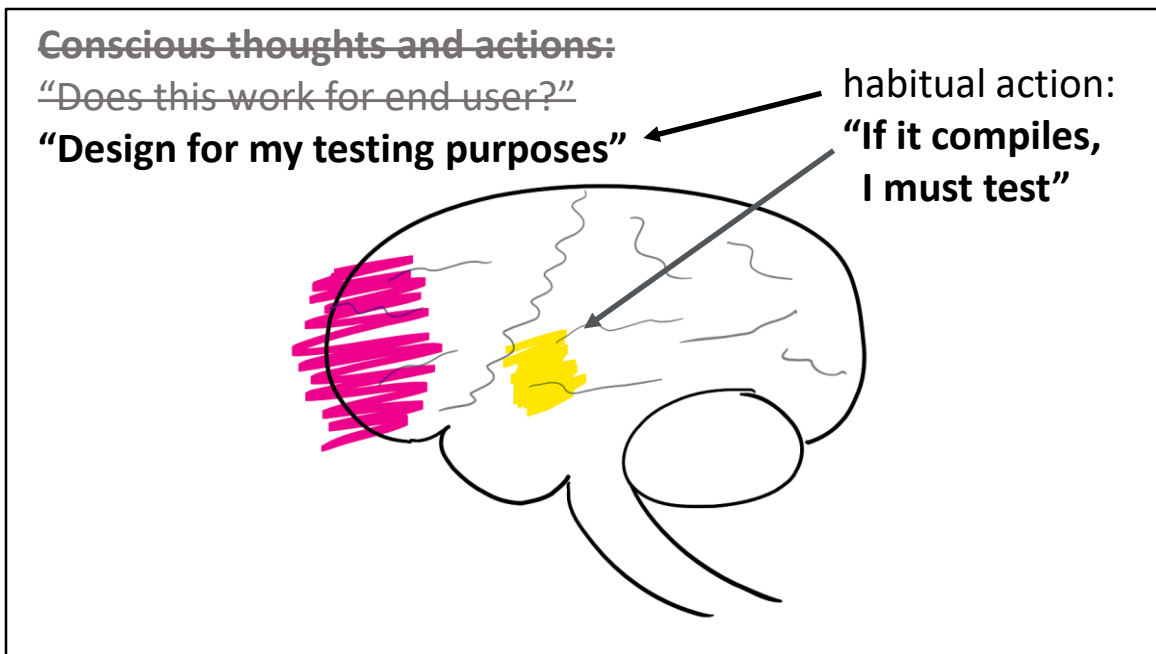
**habitual action:**



So what if...  
we're ready to test for end user scenarios



**BUT** Remember testing is a strong habit  
Just like tying our shoes, we're not conscious of what we're doing



Without being consciously aware we start design for our own testing purposes



**Now that you have read the veritas, how do you break out of dev tool extermis**

# You know what to do...

You've always known WHAT to do – that is.

This is NOT a talk on UX design

**The difference is NOW you know \*\*WHY\*\* you need to follow them.**

# Test defaults in clean state

"After we achieve a task's primary goal, we often forget cleanup steps"

**"Attention is a very scarce resource. Our brain does not waste it by keeping it focused on anything that is no longer important."**

- "Turn car headlights OFF after arrival, to prevent draining the battery"

all quotes from - Designing with the Mind in Mind p106

Back to dev tools

**- When we test our developer tools, how often do we remember to revert to an absolutely clean state. Why it is so hard to start from a clean image. Our brain is screaming, "Yay! We're done with the task!"**

- We overlook things like our default settings in our dev tools

- Don't forget to set default out of the box that are usable

- It's bad enough we ignore our own tools, but even worse when we ignore our own defaults, and hence right out of the box the apps being built are not user friendly.

# **We learn faster when risk is low**

## **- Designing with the Mind in Mind p149**

Topic #5

Do your dev tools allow users to explore risk-free? E.g. how easy can they undo or revert?

Practical application:

- Isn't it pure evil when you drag a button onto a form and then hit delete and the code-behind or other stuff created behind-the-scenes isn't deleted?

-Designing with the Mind in Mind page 149



# **Action slips** - errors of action

# **Mistakes** - errors of knowledge

# **Lapses** - errors of memory

## Topic #6

There were numerous topics I wanted to talk about that I studied at length: contextual interference, how to acquire expertise, and human error. Notice how I said “error” and not “mistake”. mistakes are a type of error, here’s why:

### 1. Action slips

- you are driving to grocery store early one Saturday morning but start going to work. You are thinking grocery store, but “auto-pilot” takes over trying to get you to go to work. You had the right intent, but the wrong motor program was used.

- takeaways for action slips:

- try using your framework under extreme stress. Go run around the block. Go say something embarrassing to your boss, then explain why. :) Whatever it takes to get you a bit flushed / heart pumping. Then try to do your talk or use your UI. You’ll find mistakes you’ve never noticed before.

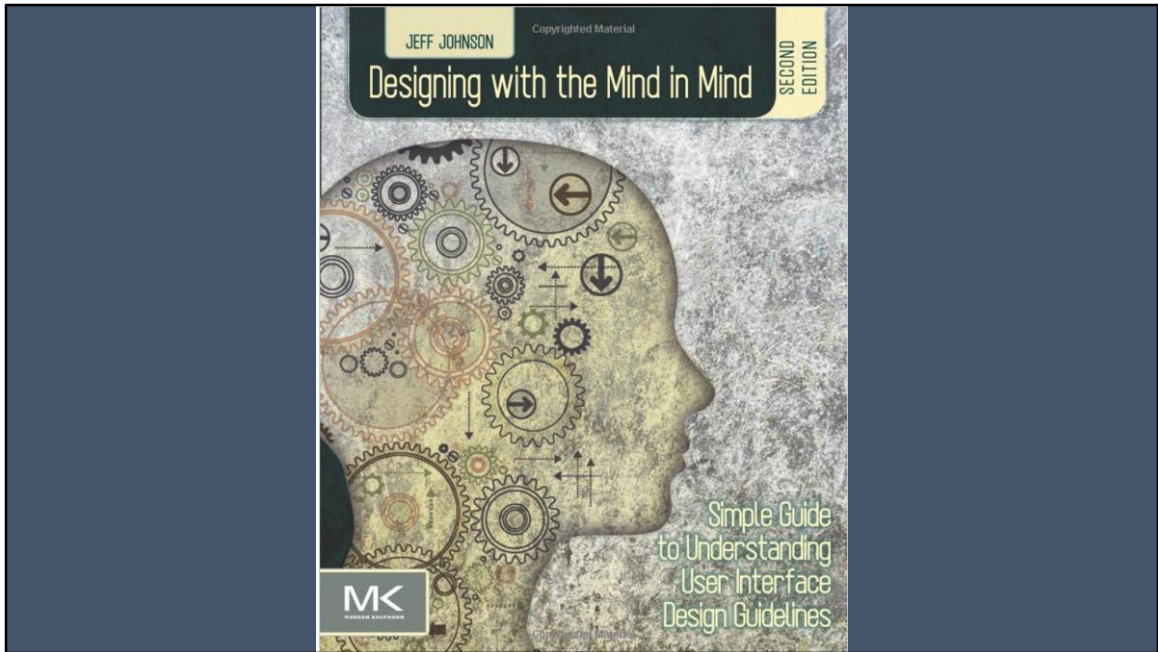
- Action slips are why I tell people to practice, practice, practice your coding demos - it’s not about when things go well. it’s all about the recovery, when things go wrong b/c of action slips. You won’t catch them until your demo becomes “auto-pilot” and you start thinking about the size of the audience, if they caught you messed up on slide 2, etc.

## 2. Mistakes

- This is where in a multiple-choice exam, I circle 'C' thinking it is the right answer, but it turns out it was 'B'. I had the right motor program, but the knowledge was wrong.
- E.g. Payton Manning said he could play as long as he did because he always knew what the defense was going to do b/c he'd seen it all before. Now compare that to rookies in the NFL. There are 6 rookie quarterbacks. They don't have that knowledge so they'll make mistakes.
- this is where new users / usability studies come in, because we know the right actions for the desired outcomes, they don't. We can't pretend not to have this knowledge. More on this in the next slide.

## 3. Lapses

- e.g. I'm seeing a ton of people I haven't seen in a while but their name escapes me. It's a lapse in memory. I know their names, i just can't remember.
- e.g. don't make the user remember the error code after you break out of a debugging session.



Tons of books, but IMO this is the best one.  
It'll save you 3.5 years of reading.

## **Seek feedback from diverse perspectives**

Break implicit biases

Remember your blue might be my green,

so when you get an unusual question – that’s your chance to figure out “what have you experienced to have you ask this question?”



**Martin LeBlanc**  
@martinleblanc

 Follow



A user interface is like a joke. If you have to explain it, it's not that good.

RETWEETS 2,811  
LIKES 1,711



10:56 AM - 14 May 2014



64



2.8K



1.7K



**<WATER BREAK>**

LABEL LABEL LABEL

**<LABEL>**

**THE ANSWER IS NEVER A LABEL**

**</LABEL>**

<https://twitter.com/martinleblanc/status/466638260195041280>

# **We're wired for language, but not for reading**

**- Designing with the Mind in Mind p149**

“Reading is unnatural.” – this is the last topic

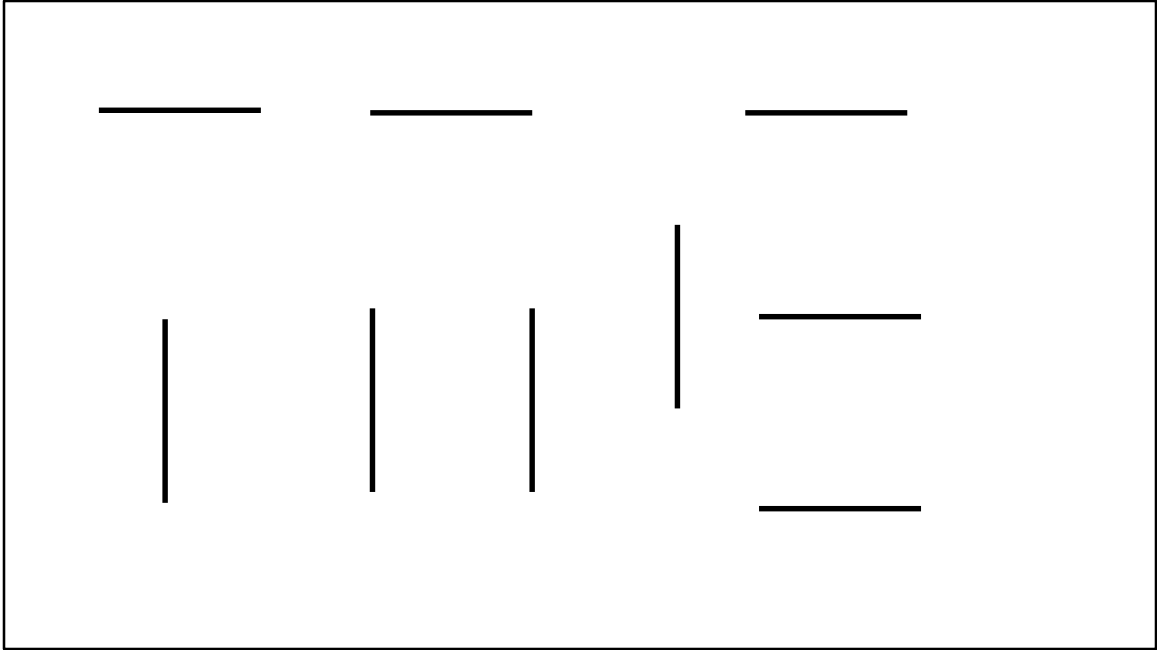
- All the piece of info below come from Designing with the Mind in Mind starting on page 33, and from every intro to psychology book out there :)

- Kids are born with an innate ability to learn spoken languages


- but as you get older it is harder to learn new languages.

- However, writing and reading are only a few thousands of years old, long after brains have evolved.

- E.g. there are very few people who never learned a spoken language, but many people never learned to read



Perception has top-down vs bottom-up processing.  
bottom-up processing - first learning to read.  
Pick up sticks



THE

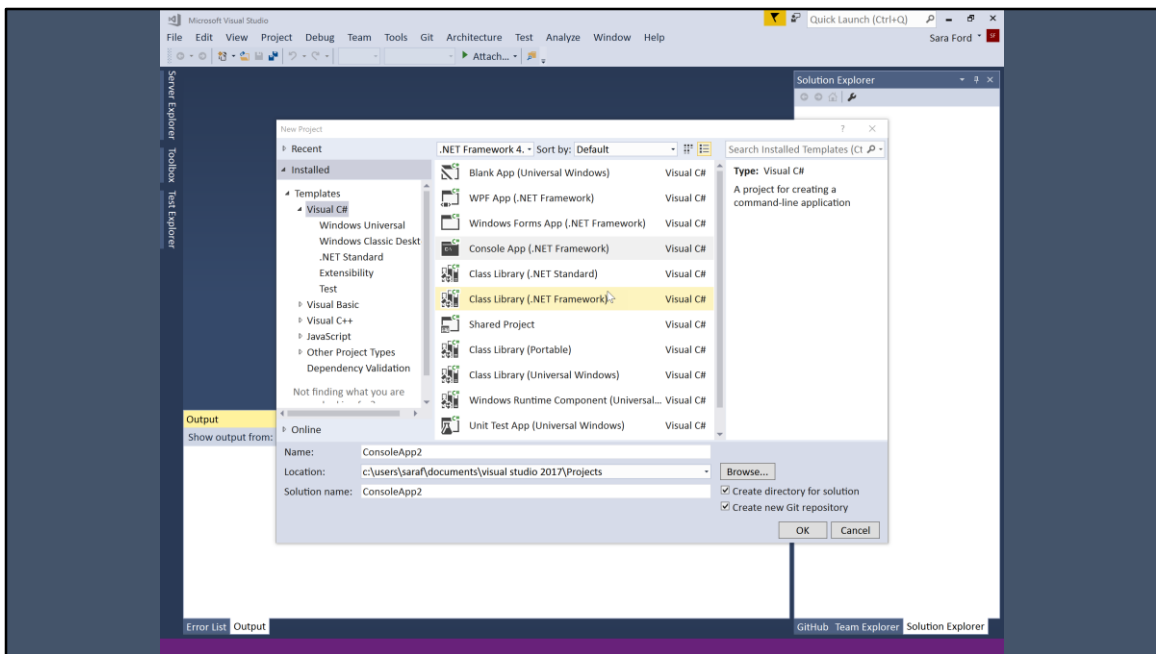
Single objects

Don't call the "letter" THE

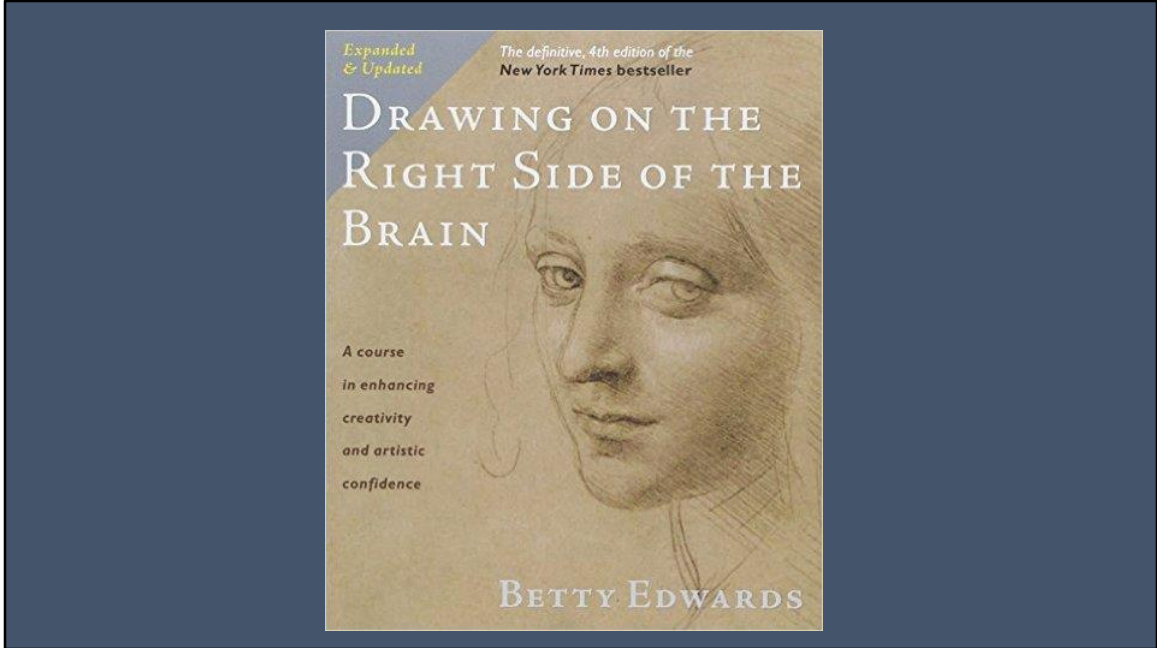
What those riddles to count 'h' in sentence

Reduces cognitive load.

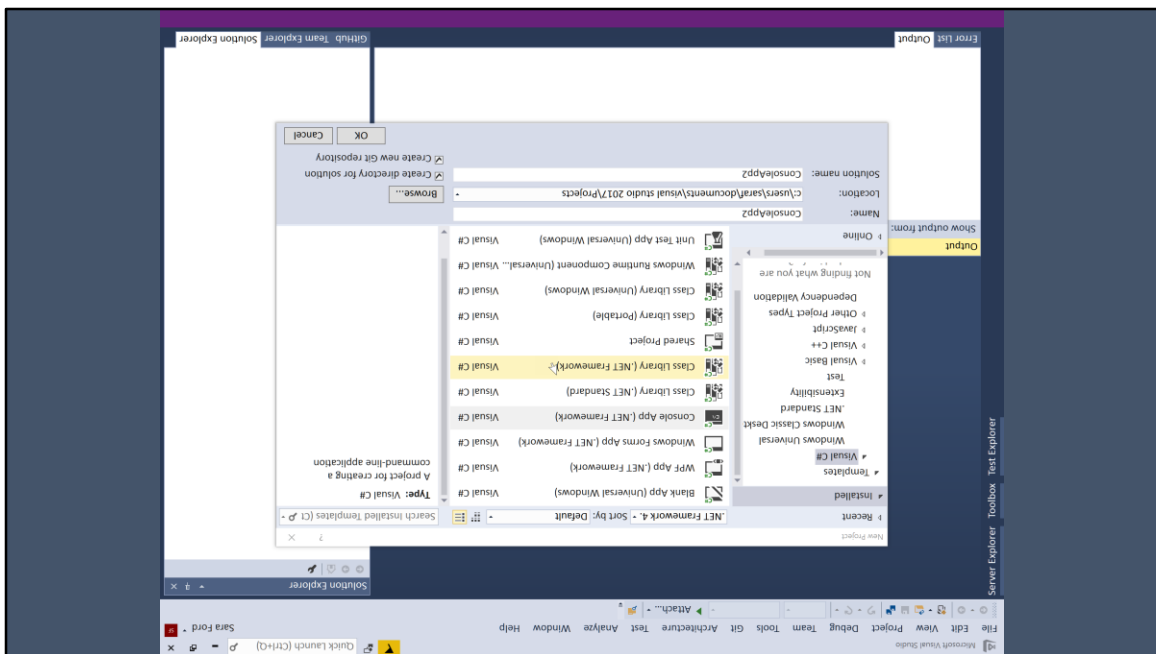




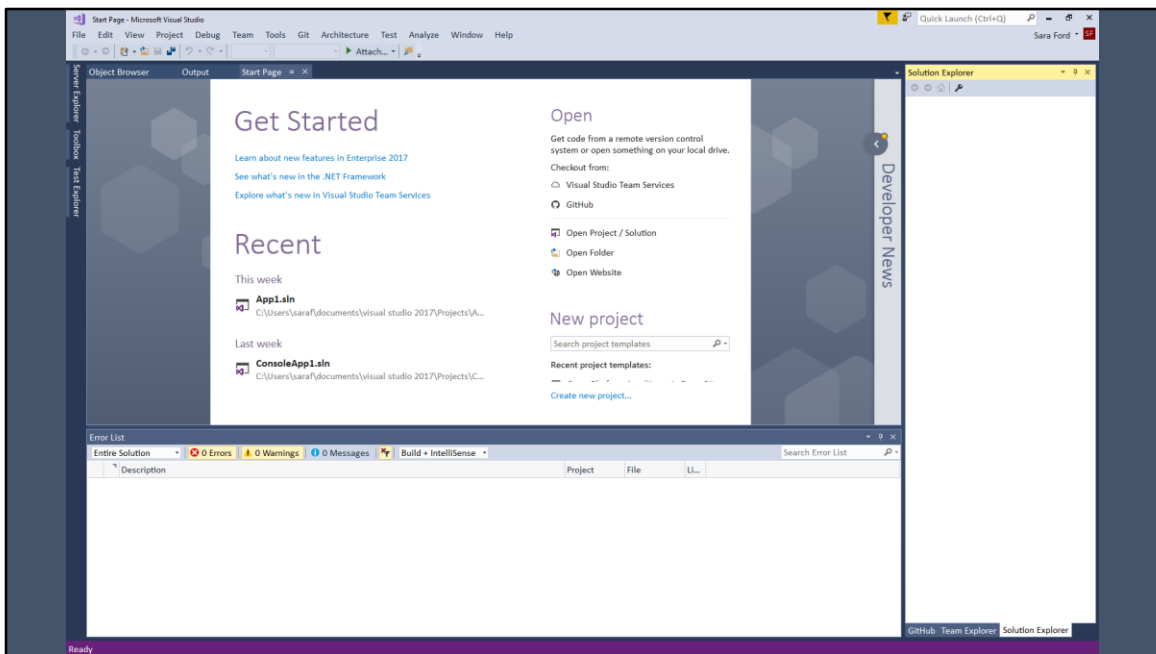
When you feel the label coming on,  
<NEXT SLIDE TO START ANIMATION>



If you want to learn how to draw, turn the page upside down, so you'll draw what is actually there.



Is this a good idea? Probably not. But at least you won't read the label!



Yes, I realize I've talked a lot about Visual Studio. It's just that being on the team...  
Adele Time!! HELLO FROM THE OUTSIDE!!!!  
**Wow, there is truly a LAST TIME for everything.**  
<Next slide>

# Just because something is complex doesn't mean it can't be usable

@sarafor

<https://sarafor.net>

Designing with the Mind in Mind

<https://git.io/DevTools> - this talk on GitHub

**Remember your take home message!**

**This is my challenge to you**

< go all in braveheart >

The next time you hear someone say "it's too complex to be usable" **you challenge that assumption.**

1. If we can make CMS usable
2. if we can make tools for medical procedures usable (please let that be true),
3. if we can make our apps that fit within a 3x5 index card usable

**WE CAN MAKE OUR DEVELOPER TOOLS USABLE**

And my challenge to myself is that I'm going to get pushing through with all my daily leg exercises and physical therapy, so the next time I thank my doctor on stage, I'm going to do it as an **international** public speaker.