



/serverless/ **DAYS**  
*Belfast*

# Preparing A Great Talk

Garth Gilmour

**DOON'T  
PANIC**

I know a bit about presenting...

---



# My Personal Code

---



**Garth Gilmour**  
@GarthGilmour



Rules for trainers:

- 1) It's not about you
- 2) Reading the play is not acting the play
- 3) It's not about you
- 4) Love what you're teaching, or at least pretend to
- 5) Write copious slides, then don't use them
- 6) Planning to live code summons the migraine fairy
- 7) Never run out of time

5:28 AM - 14 Jul 2018

# Agenda

---

- Big Ideas and General Advice
- When and how to use slides
- Final thoughts

# Part 1

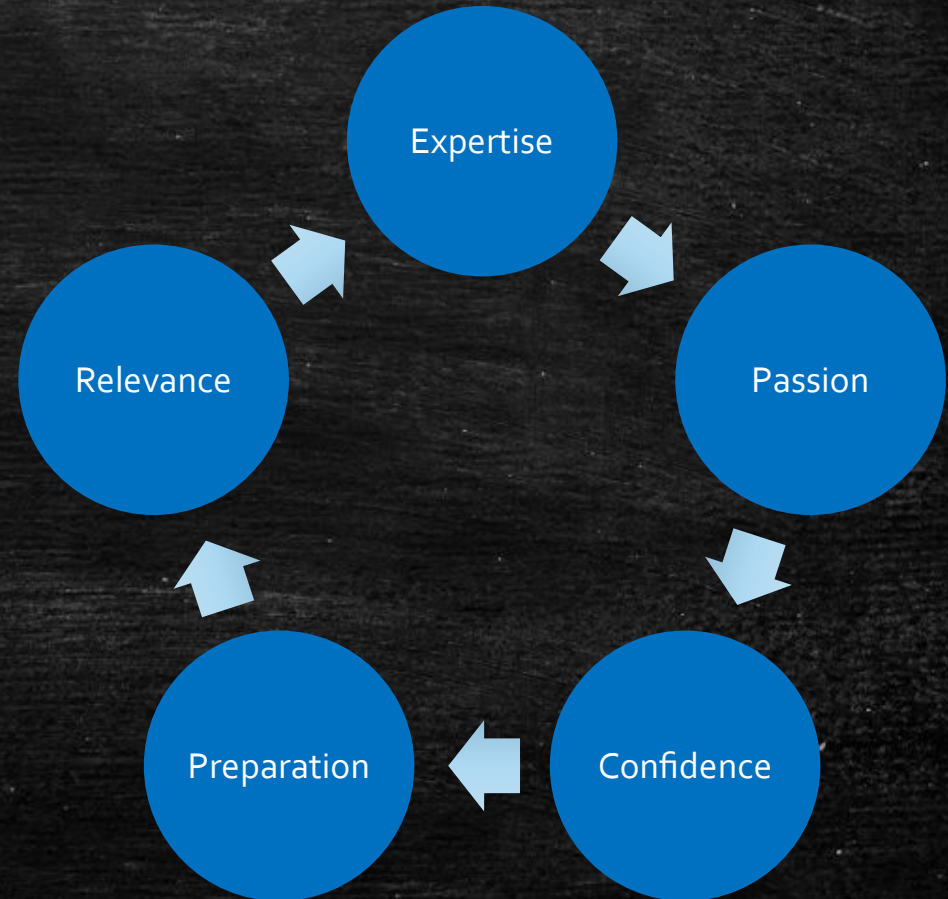
---

Big Ideas and General Advice

# The Golden Talk (which never happens)

---

- Expertise
- Passion
- Confidence
- Preparation
- Relevance



# True Story...



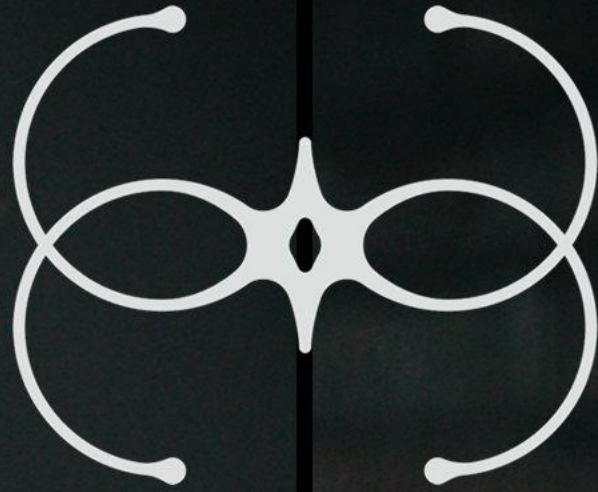


# Respect the Audience

---

- Presenting is a service industry
- The public have given time and money
- It's up to you to entertain and inform them





# Bene Gesserit

"We exist only to serve"

# Engage with the Audience

---

- The audience are on your side\*
- They will support and encourage you
- Lean into them whenever possible
- Make the most of their goodwill

\* Exceptions apply for professional speakers at commercial conferences. Plus, anyone going past their allocated time before lunch.

# Counterpoint: Take Responsibility

---

- Once you step on stage you are in the 'red shirt'
- Everything that goes wrong is your fault
- Never blame others whilst on stage



# Some Presbyterian Wisdom

---



## SERMON PREPARATION

"If only he had gone up like he went doon, then he would have gone doon like he went up..."

# Avoid Classic Blunders

---

- Too much good material (without a plan to cut it)
- Overestimating or underestimating the audience
- Assuming the audience shares your passion
- Relying on audience participation
- Relying on logistics (e.g. networks)



# Stuff Happens

---



**Garth Gilmour**

@GarthGilmour



Blinding stage lights. Speaker clock set to 45 hours, not minutes. Projector bulb blew. Fire alarm. Carbon Monoxide alarm. Fire. Computer virus. Virus. Explosive nose bleed. Attendees laid off that day. Car stolen the night before. Locked in / out of venue. Migraines and asthma.

# True Story

---

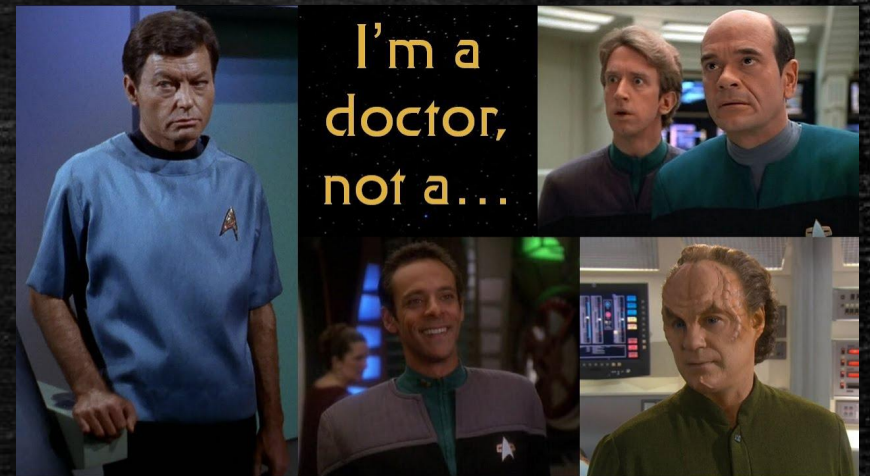




# Know Your Limitations

---

- All skills are terrifyingly context dependent
- This applies to presenting as well



## Compare the following...

---

- Lecturing students for an exam you will be setting
- Coaching developers who have to begin a new project
- Inspiring staff facing an existential threat to their jobs
- Participating in rumbustious televised political debate
- Giving a mother / father of the bride speech at a wedding

True Story...

---



# Prepare for Imposter Syndrome

---

- If this is your first talk you will hate yourself afterwards
- This is part of the process and nothing to worry about
- Don't let it bother you (if at all possible)
- Bank the lessons learned and move on



# Prepare for Imposter Syndrome

---

- It probably went a lot better than you think
- A musician notices every slip, but the audience does not...



The first pancake is never any  
good

Russian Proverb

# Be Positive

---

- Enthusiasm is contagious
- Unfortunately, so is boredom
- As the speaker you can set the tone
- Be as happy as the situation allows
  - This also applies to espionage





# On Writing

---

- Writing a good talk is bloody hard work
  - I suggest you chip away at it over time
- Give your subconscious time to percolate
  - Keep a notepad with you to jot down ideas



Writing is easy. You only need to stare at a piece of blank paper until your forehead bleeds.

Douglas Adams

# On Writing

---

- Structure is as important as content
  - Tell them what you're going to tell them
  - Tell them
  - Tell them what you just told them
- Force yourself to 'kill your darlings'
  - Only cover what the audience will find interesting
  - If you are an expert you must ruthlessly cut back

# Simplicity in Action



# Preparation

---

- In general it is not possible to 'wing it'
  - Unless you know the subject backwards and have no nerves
- There is no such thing as 'too much preparation'
  - Remember to plan for both failure and success
- Its OK to prepare things that you never use
  - Contingencies for eventualities build confidence

# Victory Loves Preparation

Unknown

# Never Knowingly Under Cater



# Spontaneity

---

- There is no such thing as too much planning / rehearsal
- Repetition can dull your edge and make you look stale...
- ... but there are ways to maintain enthusiasm



# Maintaining Enthusiasm

---



Enoch  
Powell

# Spontaneity

---

- Plan all your unplanned digressions
- Have your 'on the fly' comments prepared well in advance 😊

The hardest part is  
making it look  
spontaneous

Bono

# Adrenaline Issues

---

- Adrenaline will screw with your head
- You may become braver and mad ideas may come to mind
- But you will not become smarter or more dexterous

# Adrenaline Issues

---



We don't rise to the level of  
our expectations, we fall to  
the level of our training

Archilochos

# Be Careful With Jokes

---

- Vet all the jokes and stories you plan to tell in advance
- Humour is subjective and your talk may be shared widely

# Live Coding

---

- Live coding etc. is to be approached with great care
- Always have secure 'save points' that you can move to
- Assume all the logistics at the venue are against you



# Rehearse and Reuse

---

- No professional every does a talk just once
  - Politicians have their 'stump speech'
  - Comedians have their own 'skits'
  - Preachers save their sermons
  - Etc...
- Don't feel bad about reusing content and themes
  - As long as its new to this particular audience

# Original Content

---



— A BIT OF —  
FRY & LAURIE

We are what we repeatedly do. Excellence, then, is not an act, but a habit.

Will Durant

# Handling Questions

---



# Part 2

---

When and how to use slides

Why use slides?

---



# The Answers

---

- To make an appeal to authority
- To make a bold statement via an image
- As an anchor to talk around
- To sum up a lot of information quickly
- To simplify a complex issue
- As a fall back in case you become frozen
- For when you are too tired to think
- To be entertaining (hopefully)
- For when the demo fails
- To refer to other resources

# Making an Appeal to Authority

---

After **nine years** of work and **billions of pounds** already spent, the UK government recently abandoned a doomed IT project because **'it wasn't essential'**.

Similar examples of less epic proportions are all around us. Commercial organisations across the European Union **lost 142 billion EUR** on failed IT projects in 2004 alone, mostly because of poor alignment with business objectives or business strategies becoming obsolete during delivery.

This is roughly the cost of the **International Space Station programme**, including all flights, or almost **twice the cost of the entire Apollo programme**, which achieved six manned landings on the Moon.



# Making an Appeal to Authority

## This Is Agile Thinking



**Kent Beck** @KentBeck

20 Aug

it's not the number of hours i work in a day that measures progress, it's the number of feedback loops i complete

Expand



**Kent Beck** @KentBeck · Apr 11

1. Release to customers twice as often.
2. Fix the problems.
3. Repeat.

(Maybe release one third as much stuff at a time, just to avoid, you know, going out of business. It's okay, because much of the second two thirds your customer didn't want anyway and now you'll know early.)



5



268



439



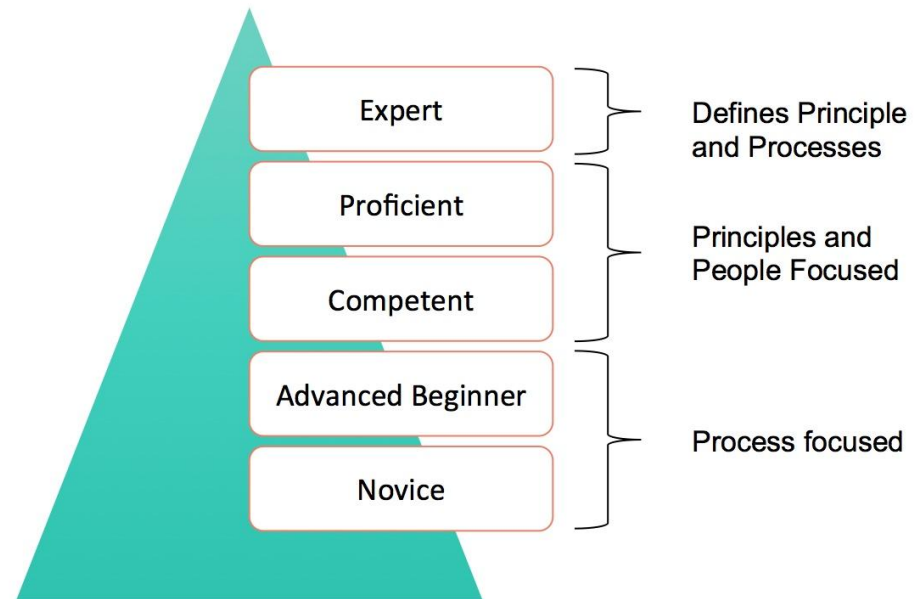
# To Make a Bold Statement via an Image

---



# As an Anchor to Talk Around

## Dreyfus Model of Skills Acquisition

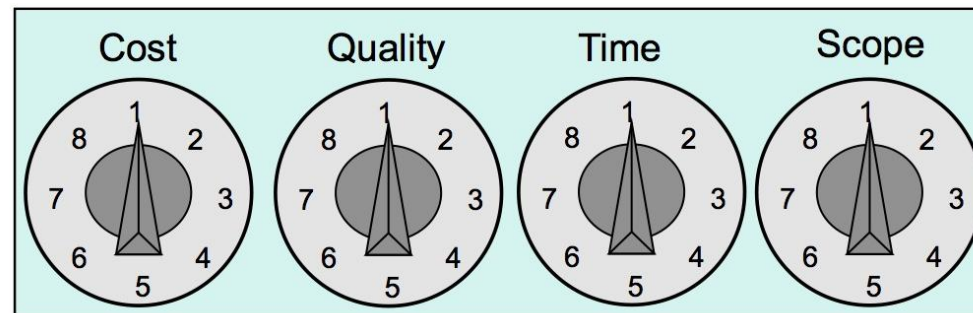


# As an Anchor to Talk Around

## Some Things Do Not Change

Some aspects of development are timeless

- Especially the relationships between the four key variables
- You can fix three of these dials but the fourth goes where it will...



# To Sum Up a Lot of Information Quickly

## The Sample Re-Written in Kotlin

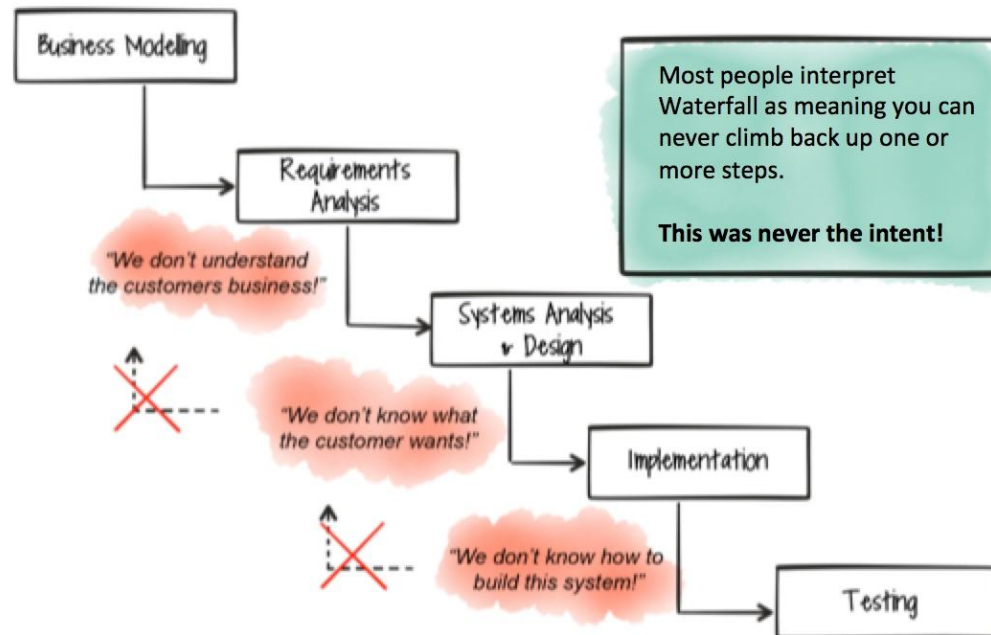
```
fun main(args: Array<String>) {  
    val numbers = mutableListOf<Int>()  
    val scanner = Scanner(System.`in`)  
    val endOfInput = Regex("X{3}")  
    println("Enter some numbers or three 'X' to finish")  
  
    while (scanner.hasNextLine()) {  
        if (scanner.hasNext(endOfInput.toPattern())) {  
            break  
        } else if (scanner.hasNextInt()) {  
            numbers += scanner.nextInt()  
        } else {  
            val mysteryText = scanner.nextLine()  
            println("Ignoring $mysteryText")  
        }  
    }  
    //Would be better to use 'numbers.sum()'  
    val total = numbers.fold(0, Int::plus)  
    println("Total of numbers is: $total")  
}
```

### Points to note:

- ✓ No redundant class
- ✓ No semi-colons
- ✓ Type inference
- ✓ Both 'val' and 'var'
- ✓ Helper functions
- ✓ String interpolation
- ✓ Simplified collections
- ✓ Interop with Java types
- ✓ Simpler use of FP

# To Sum Up a Lot of Information Quickly

## The Lack of Feedback with Waterfall



# To Simplify a Complex Issue

---

## Languages Today Are Converging



# To Simplify a Complex Issue

## The Evolution Of Web Applications



Web apps as 'browser-ware'



.NET WebForms, JEE / JSF etc...



ASP .NET MVC, Spring MVC plus  
jQuery and Dojo based widgets

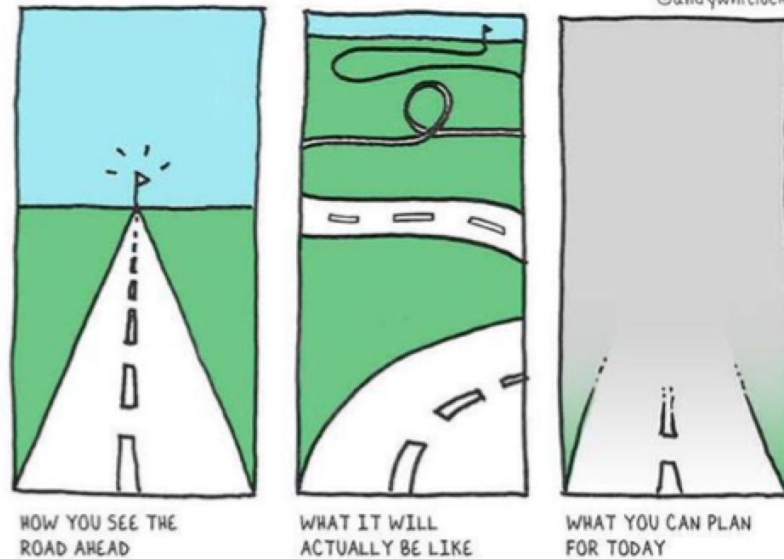


React / Angular SPA's interacting  
with RESTful Services via JSON



# To Simplify a Complex Issue

## Incremental Progress vs. Grand Design



# In Case You Freeze

---

## The Different Kinds Of Software Testing

Software is tested in many ways:

- **Functional** – ensure requirements are met
- **Usability** - UI can be understood and tasks completed
- **Reliability**
  - **Load** – ensure system operation under *expected* heavy load conditions e.g. concurrent requests, large volumes
  - **Stress** – similar to load testing but characterises the system when load is *above* expected maximum
- **Performance** - Requests are processed quickly enough
  - NB defining what 'quickly' means is essential and tough
- **Security** - application cannot be subverted or crashed
- **Supportability** - the client can install, maintain etc...

“Better a short pencil than a long  
memory”

Anonymous

“Those who advocate for single  
words on slides have never had to  
teach for eight hours with a  
\*\*\*\*\*g migraine”

Garth Gilmour

# When you are Too Tired to Think

224 **A Linked List Example**

```
graph LR
    Node1((Node)) --> Node2((Node))
    Node2 --> Node3((Node))
    Node3 --> null((null))
```

225 **Making a Start...**

When we add a new class we also add a unit test class

- An exception can be made for simple classes like JavaBeans

```
package demos.tdd;
public class LinkedList {
}

package demos.tdd;
public class LinkedListTest {
}
```

226 **The First Test**

```
package demos.tdd;
import static org.junit.Assert.*;
import org.junit.Test;

public class LinkedListTest {

    @Test
    public void newLinkedListShouldBeEmpty() {
        LinkedList list = new LinkedList();
        assertEquals(0, list.size());
    }
}
```

227 **The First Implementation**

```
package demos.tdd;

public class LinkedList {

    public boolean isEmpty() {
        return true;
    }

    public int size() {
        return 0;
    }
}
```

228 **The Second Test**

```
@Test
public void listItemsShouldNotBeEmpty() {
    LinkedList list = new LinkedList();
    list.add("abc");
    list.add("def");
    list.add("ghi");
    assertEquals(list.isEmpty());
    assertEquals(3, list.size());
}
```

229 **The Second Implementation**

```
package demos.tdd;

public class LinkedList {

    public boolean isEmpty() {
        return size == 0;
    }

    public int size() {
        return size;
    }

    public void add(String item) {
        size++;
    }

    private int size;
}
```

230 **Refactoring the Unit Test**

Our implementation is very simple

- But our test could do with some cleanup

This is a good time to refactor the test

- We can see lots of duplicated code
- This will get worse as we add more tests

It looks like every test will begin the same way

- With `LinkedList list = new LinkedList();`
- So lets move this to a method called before each test
- The test variable can become a test

Many tests will require a populated list

- So lets extract a method for loading test items

231 **Refactoring the Unit Test**

```
public class LinkedListTest {

    public void setUp() {
        list = new LinkedList();
    }

    @Test
    public void newLinkedListShouldBeEmpty() {
        assertEquals(list.isEmpty());
        assertEquals(0, list.size());
    }

    @Test
    private void addItem() {
        list.add("abc");
        list.add("def");
        list.add("ghi");
    }

    private LinkedList list;
}
```

232 **The Third Test**

```
@Test
public void counterLevelsFrom() {
    list.add("xyz");
    assertEquals("xyz", list.get(0));
}
```

233 **The Node (Obvious Code)**

```
class Node {
    Node(Node next, Node prev, String item) {
        this.next = next;
        this.prev = prev;
        this.item = item;
    }

    Node getNext() { return next; }
    void setNext(Node next) { this.next = next; }
    String getItem() { return item; }

    private Node next;
    private Node prev;
    private String item;
}
```

234 **The Third Implementation**

```
public void add(String item) {
    if(isEmpty()) {
        first = new Node(null, null, item);
        size++;
    }

    public String get(int index) {
        if(index == 0) {
            return first.getItem();
        }
        return null;
    }

    private Node first;
}
```

235 **The Fourth Test**

```
@Test
public void counterLevelsMultipleTimes() {
    addItem();
    assertEquals("abc", list.get(0));
    assertEquals("def", list.get(1));
    assertEquals("ghi", list.get(2));
}
```

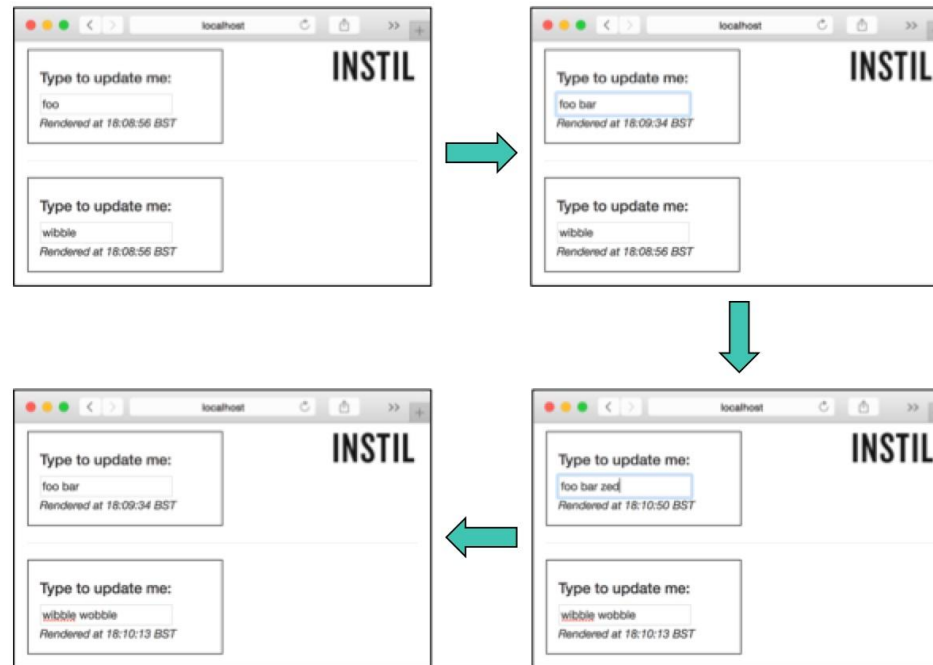
# To Be Entertaining (Hopefully)

## When It All Goes Wrong...



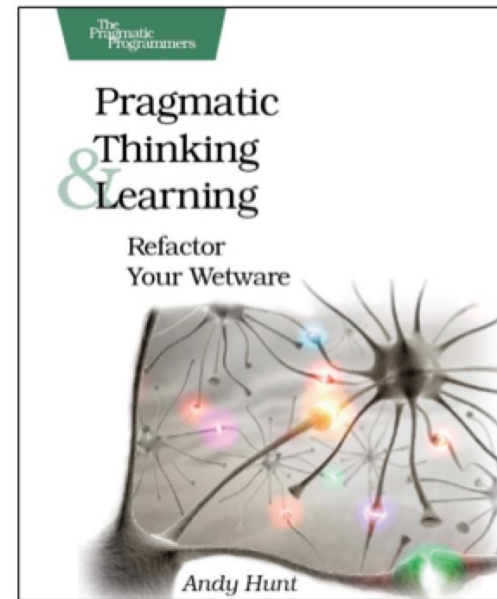
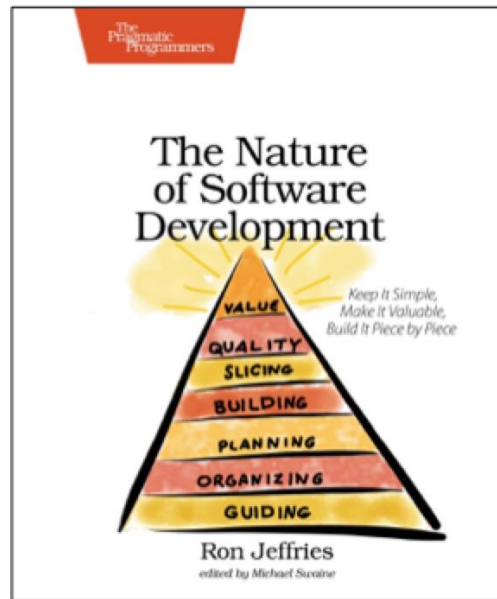
# For When the Demo Fails

## Viewing Changes in the DOM



# To Refer to Other Resources

---



# Part 3

---

Final Thoughts



# An Alternative View

The screenshot shows a YouTube video player with a presentation slide. The slide has a teal background with white clouds at the bottom. In the top left corner, there is a Twitter icon and the handle '@LondonGophers'. In the top right corner, there is a 'GO' logo. The main title of the slide is 'It's Not All Talk' in large white font. Below the title is a circular profile picture of Benjamin Bryant and his name 'Benjamin Bryant' in white text. On the right side of the slide, there is a circular logo for 'LONDON GOPHERS' featuring a cartoon gopher wearing a brown hat and a red and white striped scarf. Below this logo is the 'SaltPay' logo in white text. The video player interface includes a progress bar at the bottom showing '0:01 / 14:00', a play button, a volume icon, a settings gear icon, and other standard video controls. A 'Settings' button is also visible in the bottom right corner of the video frame. The video title at the bottom of the player reads 'It's Not All Talk - Benjamin Bryant - November Gophers 2022'.

<https://www.youtube.com/watch?v=B1SoXbgdgKs>

# My Personal Code (Revisited)

---



**Garth Gilmour**  
@GarthGilmour



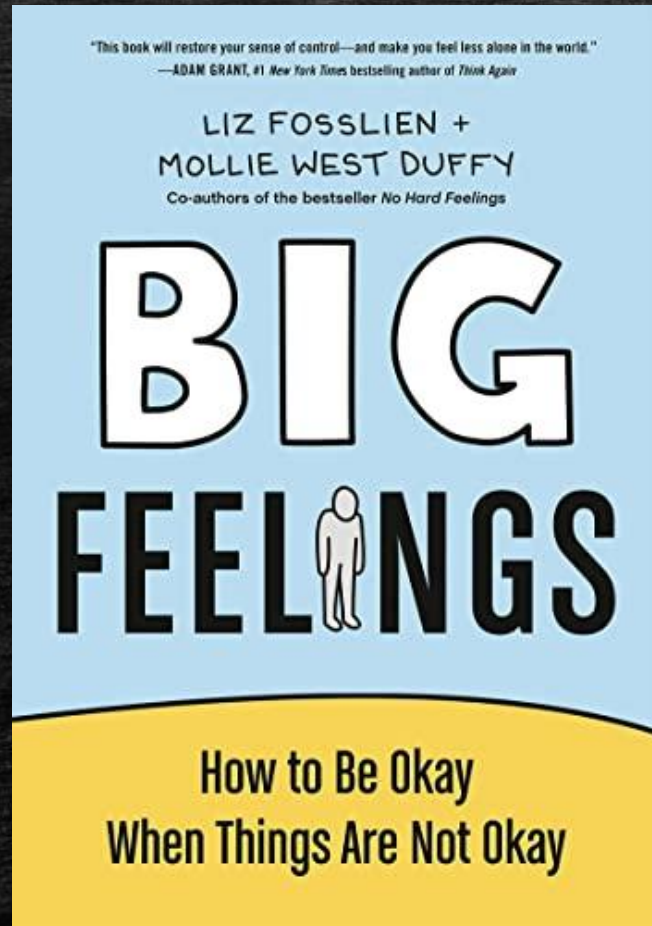
Rules for trainers:

- 1) It's not about you
- 2) Reading the play is not acting the play
- 3) It's not about you
- 4) Love what you're teaching, or at least pretend to
- 5) Write copious slides, then don't use them
- 6) Planning to live code summons the migraine fairy
- 7) Never run out of time

5:28 AM - 14 Jul 2018

It's OK to be Nervous!

---



Questions?

