



Presenting research: *Structure, story, and support*

Bogdan Vasilescu
SSSG 1/23/2017

Slides from:
Felienne Hermans
Simon Peyton Jones

Research is communication



The greatest ideas are
worthless if you keep them to
yourself.

Public speaking is not easy



Top ten fears

- | | |
|----------------------------------|-----|
| 1. Snakes | 51% |
| 2. Speaking in public | 40% |
| 3. Heights | 36% |
| 4. Being closed in a small space | 34% |
| 5. Spiders and insects | 27% |
| 6. Needles and getting shots | 21% |
| 7. Mice | 20% |
| 8. Flying on a plane | 18% |
| 9. Dogs (sorry, Lassie) | 11% |
| 9. Thunder and lightning | 11% |
| 9. Crowds | 11% |
| 10. Going to the doctor | 9% |

*Gallup Poll, February 18-21, 2001
(1,016 respondents + or - 3%)

Good news: only one rule to presenting



A photograph showing two men from the chest up, looking down at a newspaper they are holding together. The man on the left is wearing glasses and has white hair. The man on the right has dark hair and is wearing a suit jacket. They appear to be in an indoor setting.

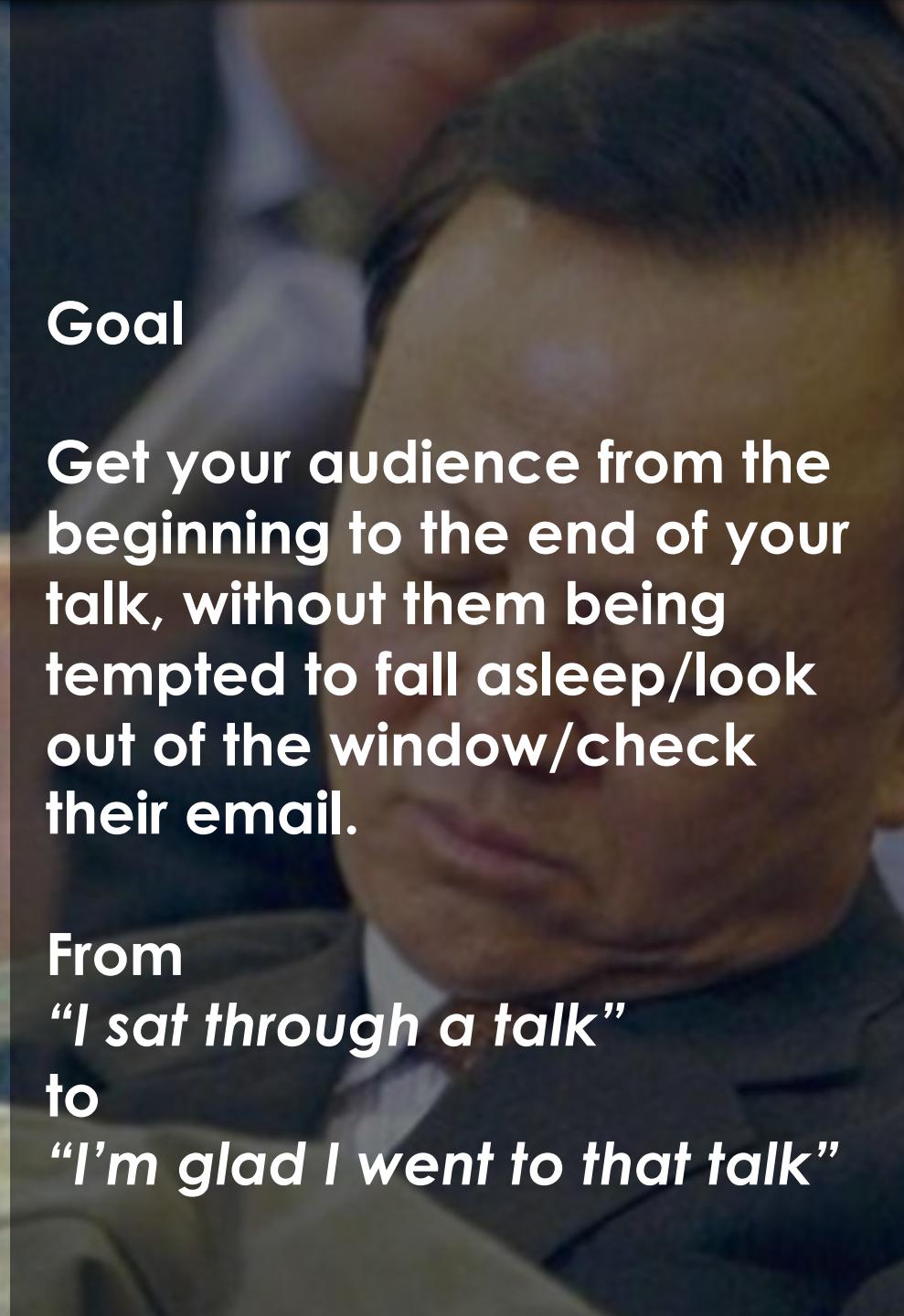
**Don't lose
your audience**



**Don't lose
your audience**

Goal

Get your audience from the beginning to the end of your talk, without them being tempted to fall asleep/look out of the window/check their email.



From
"I sat through a talk"
to
"I'm glad I went to that talk"



**Don't lose
your audience**



Put audience first!

A research talk gives you access to an invaluable commodity: the time and attention of other people. Don't waste it!

This presentation gives you tools for getting your audience through your talk.



In other words:

minimize 'exit moments',
moments where the
audience might drop out.



**Don't lose
your audience**

1) Right structure

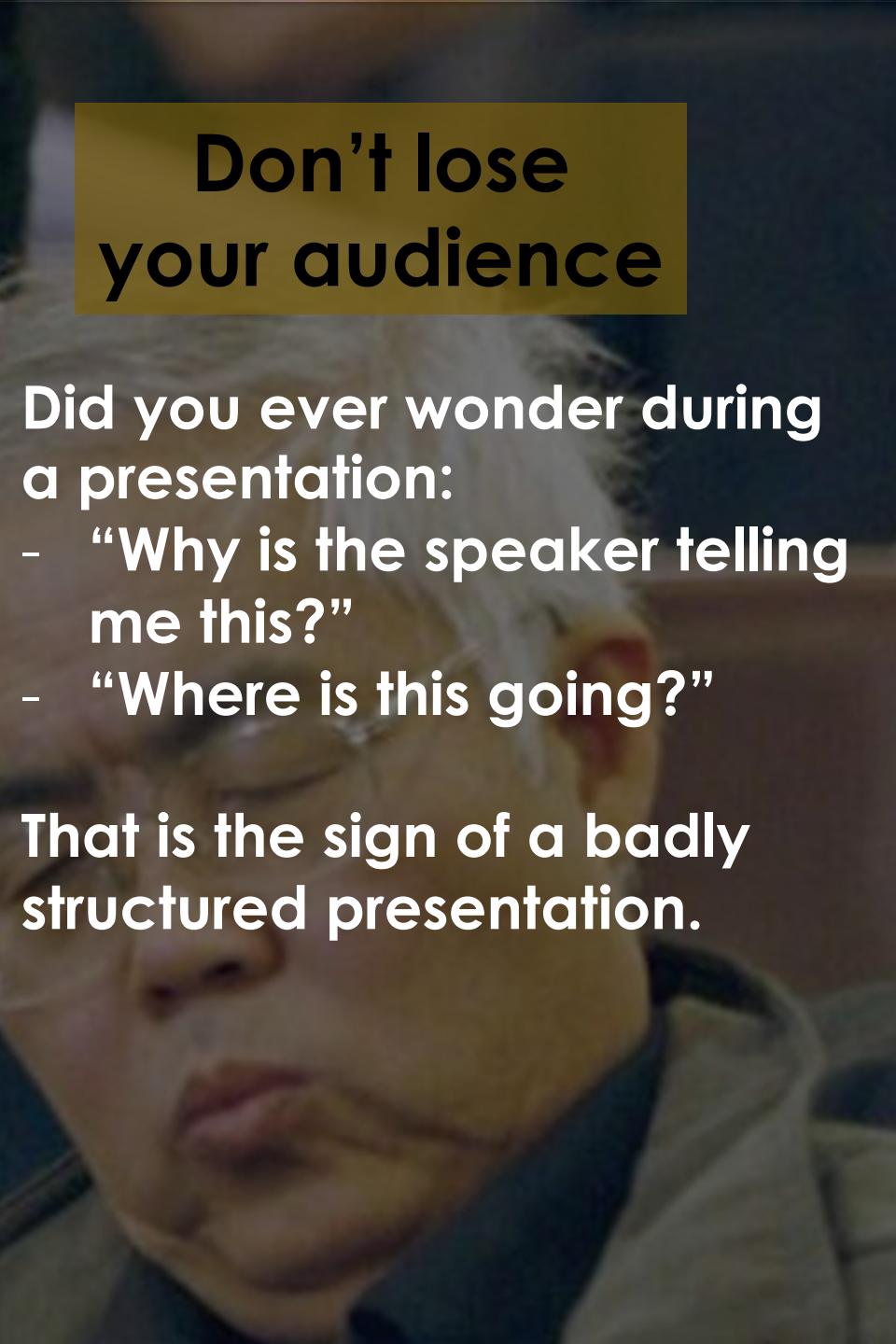
2) Adding stories

3) Support

A photograph of two men looking down at a newspaper. On the left, an older man with glasses and white hair is visible. On the right, another man with a shaved head is also looking down at the paper. They appear to be in a public setting, possibly a train or bus, as a window and other passengers are visible in the background.

**Don't lose
your audience**

1) Right structure

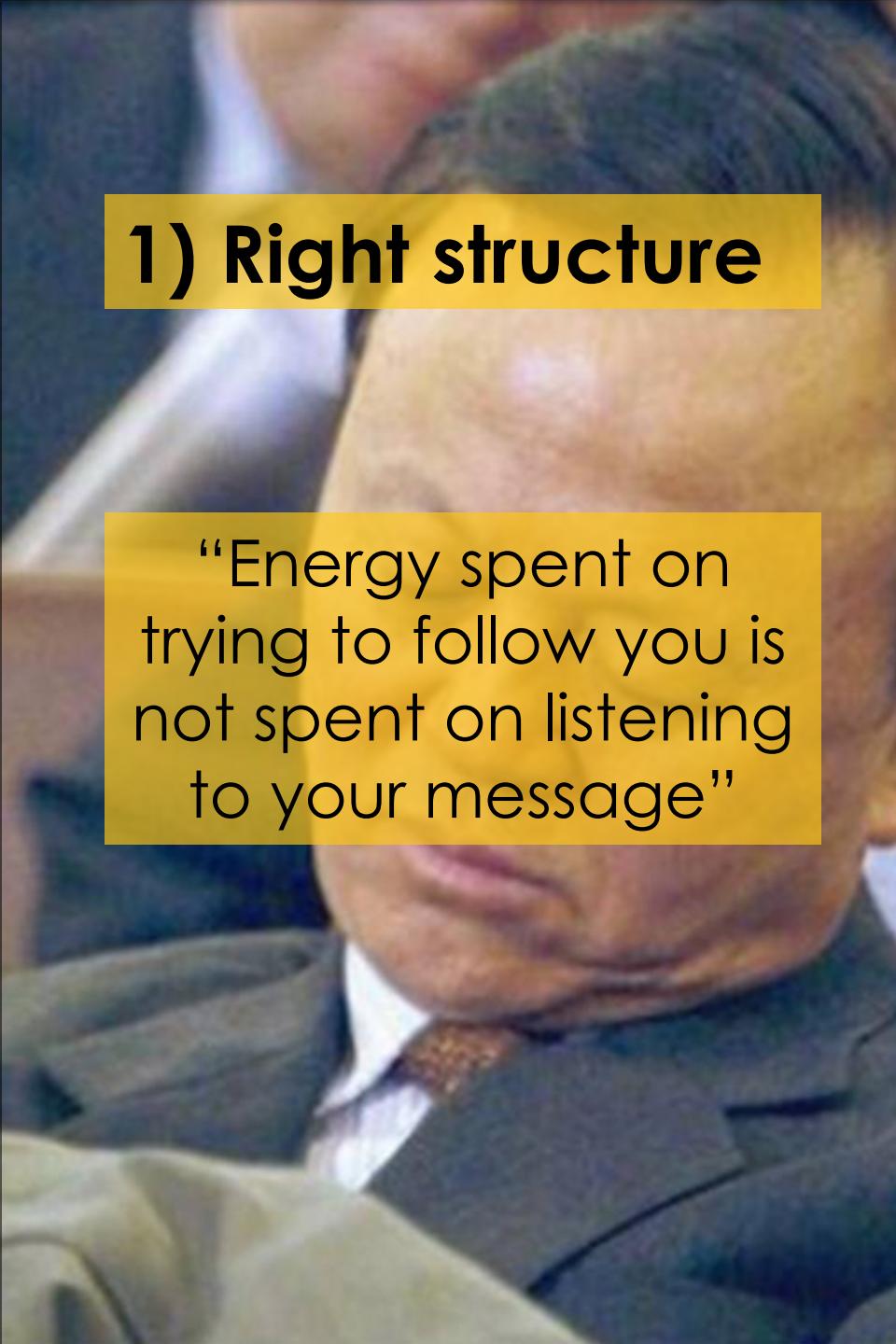


Don't lose your audience

Did you ever wonder during
a presentation:

- “Why is the speaker telling
me this?”
- “Where is this going?”

That is the sign of a badly
structured presentation.

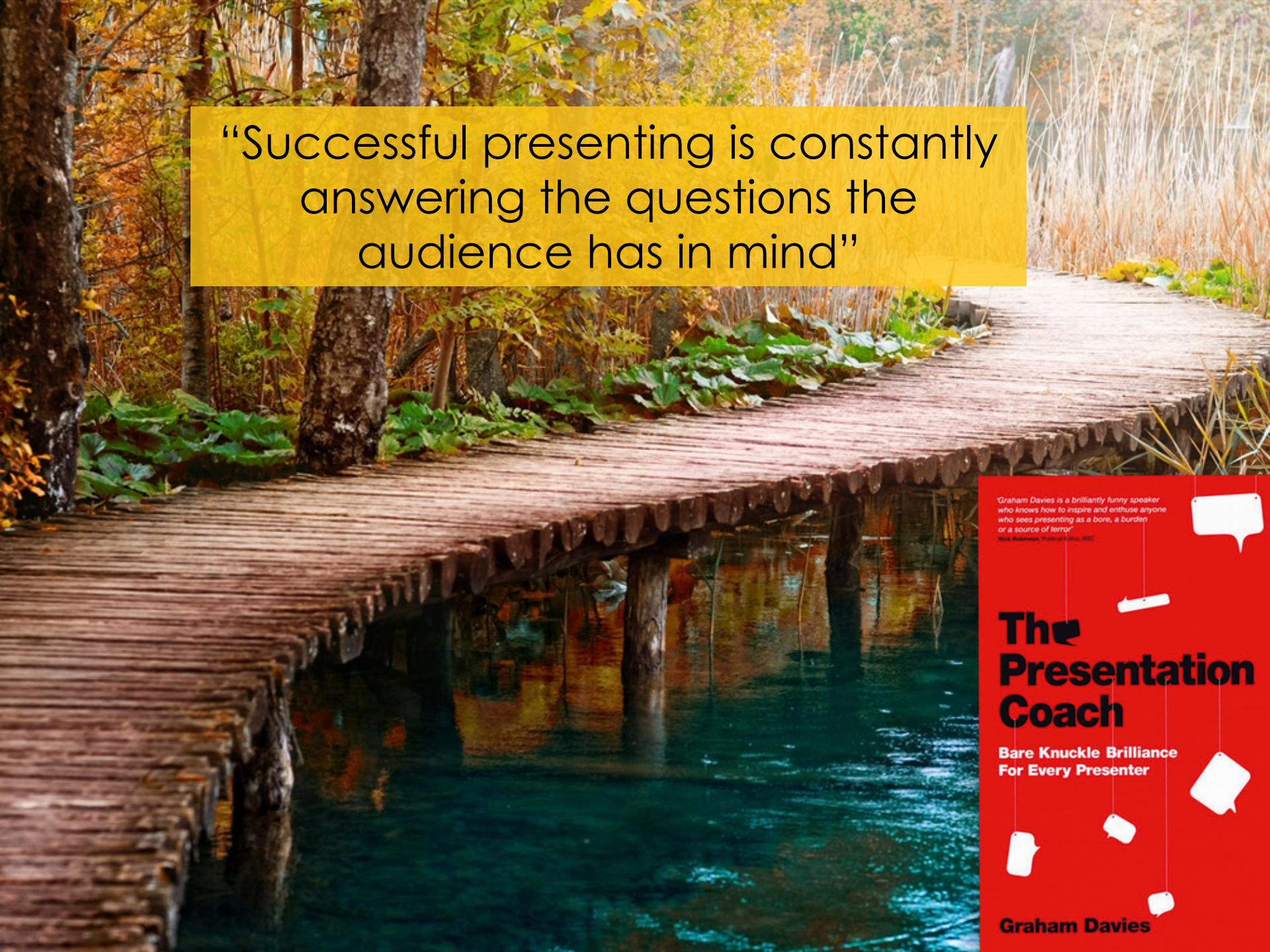


1) Right structure

“Energy spent on
trying to follow you is
not spent on listening
to your message”

Structure gets your audience from A to B





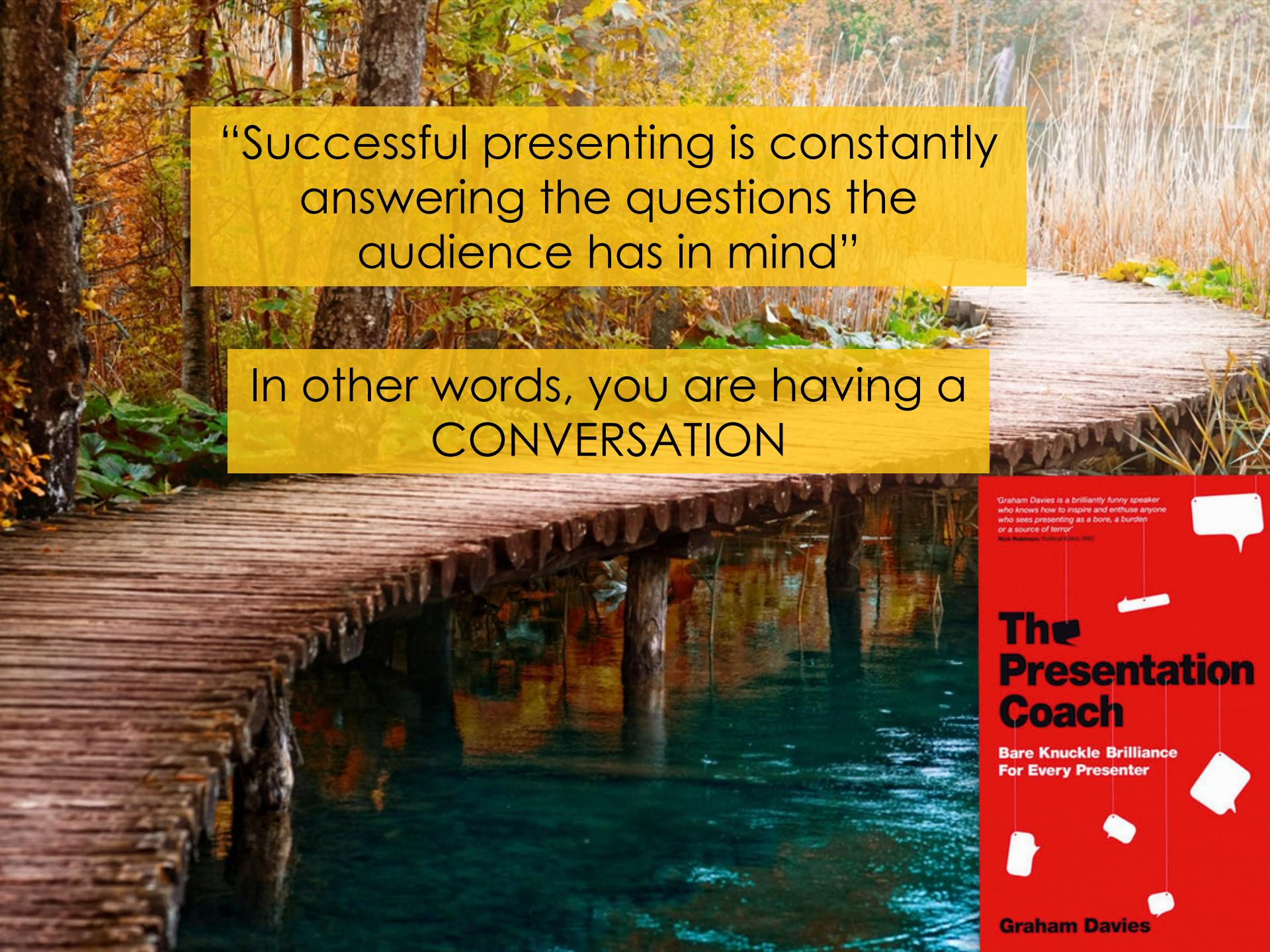
“Successful presenting is constantly answering the questions the audience has in mind”

‘Graham Davies is a brilliantly funny speaker who knows how to inspire and enthuse anyone who sees presenting as a bore, a burden or a source of terror’
Nick Robinson, Political Editor, BBC

The Presentation Coach

Bare Knuckle Brilliance
For Every Presenter

Graham Davies



“Successful presenting is constantly answering the questions the audience has in mind”

In other words, you are having a CONVERSATION

Graham Davies is a brilliantly funny speaker who knows how to inspire and enthuse anyone who sees presenting as a bore, a burden or a source of terror!
Nick Robinson, Political Editor, BBC

The Presentation Coach

Bare Knuckle Brilliance
For Every Presenter

Graham Davies

A black and gold microphone stands on a silver tripod in the foreground, positioned to the right of the text. The background is a blurred image of a theater interior with wooden seating and a large audience.

**What people think about
when they prepare a talk**



**What you should think about
when preparing a talk**

What kind of work do you do?



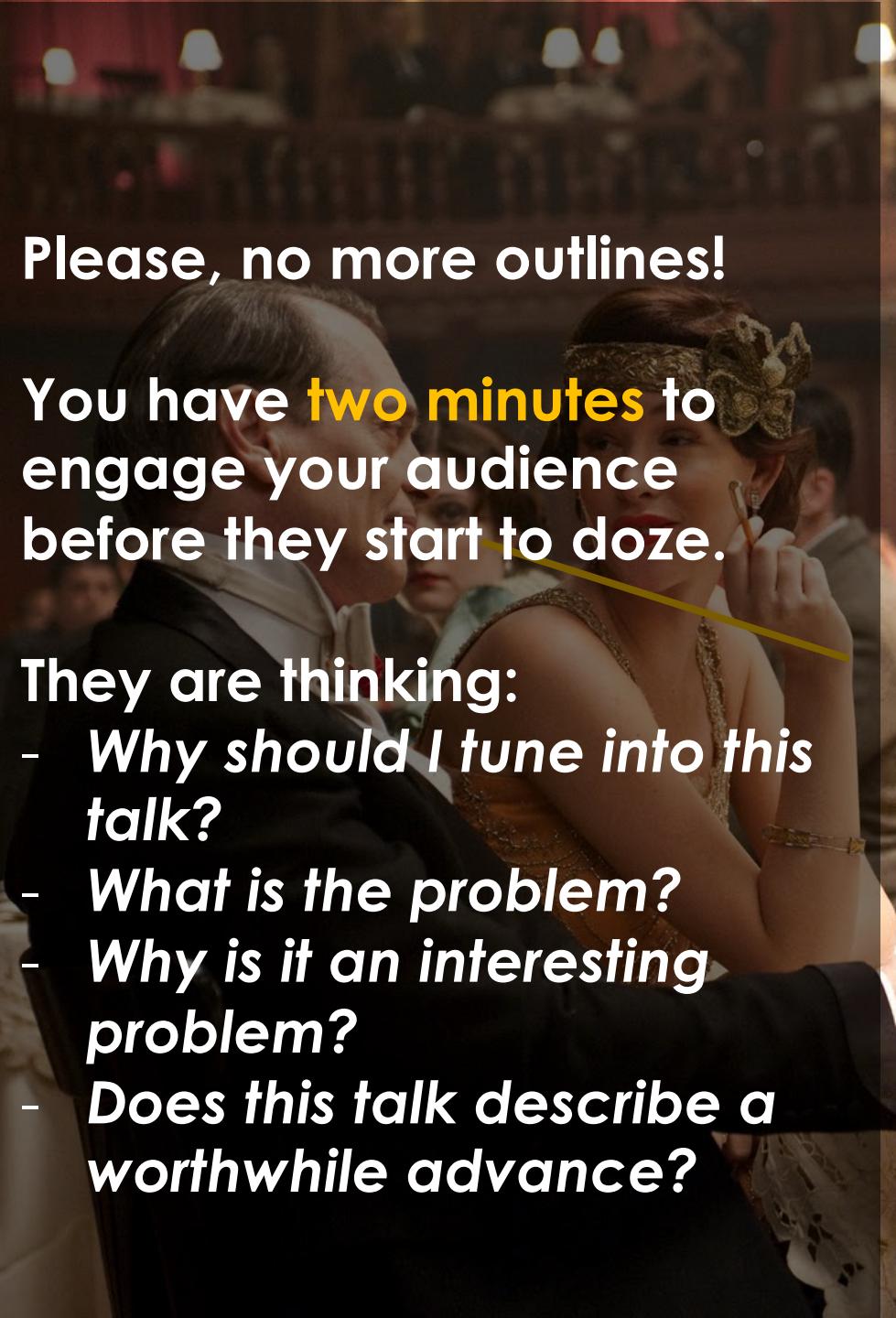
What kind of work do you do?

A scene from the movie "The Great Gatsby". A man in a dark tuxedo and white shirt is looking towards a woman. She is wearing a gold sequined dress, a large ornate headband, and a pearl necklace. She is holding a cigarette holder. They are seated at a table with a lace tablecloth, surrounded by other people in a formal dining room.

First I will tell you how local government works in the US

Then I will elaborate on the position of city treasurer

Finally, I will tell you about some legal troubles I am currently facing.



Please, no more outlines!

You have **two minutes** to engage your audience before they start to doze.

They are thinking:

- ***Why should I tune into this talk?***
- ***What is the problem?***
- ***Why is it an interesting problem?***
- ***Does this talk describe a worthwhile advance?***



First I will tell you how local government works in the US

Then I will elaborate on the position of city treasurer

Finally, I will tell you about some legal troubles I am currently facing.

A black and white photograph of a man and a woman in 1920s-style formal wear seated at a round table in a restaurant. The man, on the left, is wearing a dark tuxedo and a white shirt with a bow tie. The woman, on the right, is wearing a light-colored dress with gold embroidery and a large, ornate headband. They are looking towards each other across the table. The background shows other diners and waitstaff in a dimly lit restaurant.

Think about
conversation
if you structure a
presentation

What to put in?

1

Motivation
(20%)

2

Your key
idea (80%)

3

There is
no 3

Why should I tune
into this talk?

“If you remember
nothing else,
remember this.”

Some ideas



Some ideas

Start with the end:
“50% more performance”



Some ideas

Start with the end:
“50% more performance”

Example: Java class files
are large (brief figures), and
get sent over the network.
Can we use language-
aware compression to
shrink them? Yes, and I’m
going to show you how we
can do 50% better than the
best generic zipping
technology.



The Sky Is Not the Limit: Multitasking Across GitHub Projects

Bogdan Vasilescu[†], Kelly Blincoe[§], Qi Xuan[‡], Casey Casalnuovo[†], Daniela Damian[‡], Premkumar Devanbu[†], Vladimir Filkov[†]

[†]Dept. Computer Science, University of California, Davis, Davis, CA 95616, USA
[‡]Dept. Electrical & Computer Engineering, University of Auckland, New Zealand

[§]Dept. Automation, Zhejiang University of Technology, Hangzhou 310023, China
[†]Dept. Computer Science, University of Victoria, Victoria, BC, Canada

{vasilescu, ccasal, pdevanbu, vfilkov}@ucdavis.edu
k.blincoe@uckland.ac.nz, xuanqi@zjut.edu.cn, danielad@uvic.ca

ABSTRACT

Software development has always inherently required multitasking: developers switch between coding, reviewing, testing, designing, and meeting with colleagues. The advent of software ecosystems like GitHub has enabled something new: the ability to easily switch between projects. Developers also have social incentives to contribute to many projects; prolific contributors gain social recognition and (eventually) economic rewards. Multitasking, however, comes at a cognitive cost: frequent context-switches can lead to distraction, sub-standard work, and even greater stress. In this paper, we gather ecosystem-level data on a group of programmers working on a large collection of projects. We develop models and methods for measuring the rate and breadth of a developers' context-switching behavior, and we study how context-switching affects their productivity. We also survey developers to understand the reasons for and perceptions of multitasking. We find that the most common reason for multitasking is interrelationships and dependencies between projects. Notably, we find that the *rate of switching* and *breadth* (number of projects) of a developer's work matter. Developers who work on many projects have higher productivity if they focus on few projects per day. Developers that switch projects too much during the course of a day have lower productivity as they work on more projects overall. Despite these findings, developers' perceptions of the benefits of multitasking are varied.

CCS Concepts

•Information systems → Data analytics; •Human-centered computing → Empirical studies in collaborative and social computing; •Software and its engineering → Open source model;

Keywords

Multitasking; GitHub; productivity

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICSE '16, May 14 - 22, 2016, Austin, TX, USA
© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-3900-1/16/05...\$15.00
DOI: <http://dx.doi.org/10.1145/2884781.2884875>

In any case

1. INTRODUCTION

Multitasking is a staple of high performing professionals, programmers included. It is the ability to stop working on a task, switch to another, and return eventually to the first one, as needed or as scheduled. The goal is to optimize human resource allocation, while reprioritizing tasks dynamically [17]. When done well, or at least in a disciplined way, multitasking can yield dividends [1, 4]. Clearly, if a task in the queue has a higher priority than the current one, switching them can improve performance. For example, programmers encounter this when starting a new coding task, while their previous, urgently needed fix undergoes testing. If testing uncovers bugs, the urgent fix will require attention, and the new coding task will be put on the back-burner.

Multitasking comes at a cost though [9]. Humans, programmers included, have a certain, limited amount of *cognitive flexibility*, the mental ability to switch from thinking about one concept to thinking about another. Limitations apply to both the number of concepts we can juggle, as well as to the difficulty in switching between them. For example, coding simultaneously 7 simple functions is easier than coding 8 of them. Likewise, switching between two small matrix multiplication problems is more difficult than switching between two small integer additions. As we can imagine, reaching our innate limitations can result in decreased performance on all tasks and perhaps even diminished quality. It is unknown how far multitasking can be pushed safely, although some anecdotal evidence is available.

Software developers have long been pushing the limits of multitasking [27] because of the innate modularity of the development process and the independence of module processing (*e.g.*, one can code while tests are being executed). In open-source software, developers also commonly contribute to multiple projects in the same time period, bridging different communities [20, 30]. With the advent of social coding tools like GitHub, this has intensified. It is not uncommon to find prolific developers contributing code to 5-10 GitHub projects in the same week. In fact, contributing to as many GitHub projects as possible is an accomplishment, valued by peers and employers alike [32].

There are various reasons why developers are more prolific on GitHub compared to other platforms. The features and usability provided by GitHub play a big role [34]. It is one example of how novel technology benefits programmers, and it empowers them: GitHub's platform is responsible for the inception of many projects, that otherwise wouldn't have existed [34]. And the payoffs are also substantial [34]. With

The Sky Is Not the Limit: Multitasking Across GitHub Projects

Bogdan Vasilescu[†], Kelly Blincoe[§], Qi Xuan[‡], Casey Casal-Hayot[†], Daniela Damian[‡], Premkumar Devanbu[†], Vladimir Filkov[†]

[†]Dept. Computer Science, University of California, Davis, Davis, CA 95816, USA

[‡]Dept. Electrical & Computer Engineering, University of Auckland, New Zealand

[§]Dept. Automation, Zhejiang University, Hangzhou 310023, China

[¶]Dept. Computer Science, University of Victoria, Victoria, BC, Canada

{vasilescu, ccasal, ptdevanbu, yfilkov}@ucdavis.edu

k.blincoe@uckland.ac.nz, xuanqi@zjut.edu.cn, danielad@uvic.ca

ABSTRACT

Software development has always inherently required multitasking: developers switch between coding, reviewing, testing, designing, and meeting with colleagues. The advent of software ecosystems like GitHub has enabled something new: the ability to easily switch between projects. Developers also have social incentives to contribute to many projects; prolific contributors gain social recognition and (eventual) economic rewards. Multitasking, however, comes at a cognitive cost: frequent context-switches can lead to distraction, sub-standard work, and even greater stress. In this paper, we gather ecosystem-level data on a group of programmers working on a large collection of projects. We develop models and methods for measuring the rate and breadth of developers' context-switching behavior, and we study how context-switching affects their productivity. We also survey developers to understand the reasons for and perceptions of multitasking. We find that the most common reason for multitasking is interrelationships and dependencies between projects. Notably, we find that the *rate of switching and breadth* (number of projects) of a developer's work matter. Developers who work on many projects have higher productivity if they focus on few projects per day. Developers that switch projects too much during the course of a day have lower productivity as they work on more projects overall. Despite these findings, developers' perceptions of the benefits of multitasking are varied.

CCS Concepts

•Information systems → Data analytics; •Human-centered computing → Empirical studies in collaborative and social computing; •Software and its engineering → Open source model;

Keywords

Multitasking; GitHub; productivity

Permission to make digital or hard copies of all or part of this work for personal classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICSE '16, May 14 - 22, 2016, Austin, TX, USA

© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ISBN 978-1-4503-3900-1/16/05...\$15.00

DOI: <http://dx.doi.org/10.1145/2884781.2884875>

The Sky Is Not the Limit: Multitasking Across GitHub Projects

Don't copy the paper

Bogdan Vasilescu[†], Kelly Blincoe[§], Qi Xuan[‡], Casey Casal-Hayot[†], Daniela Damian[‡], Premkumar Devanbu[†], Vladimir Filkov[†]

[†]Dept. Computer Science, University of California, Davis, Davis, CA 95816, USA

[‡]Dept. Electrical & Computer Engineering, University of Auckland, New Zealand

[§]Dept. Automation, Zhejiang University of Technology, Hangzhou 310023, China

[¶]Dept. Computer Science, University of Victoria, Victoria, BC, Canada

{vasilescu, ccasal, ptdevanbu, yfilkov}@ucdavis.edu

k.blincoe@uckland.ac.nz, xuanqi@zjut.edu.cn, danielad@uvic.ca

ABSTRACT

Software development has always inherently required multitasking: developers switch between coding, reviewing, testing, designing, and meeting with colleagues. The advent of software ecosystems like GitHub has enabled something new: the ability to easily switch between projects. Developers also have social incentives to contribute to many projects; prolific contributors gain social recognition and (eventual) economic rewards. Multitasking, however, comes at a cognitive cost: frequent context-switches can lead to distraction, sub-standard work, and even greater stress. In this paper, we gather ecosystem-level data on a group of programmers working on a large collection of projects. We develop models and methods for measuring the rate and breadth of developers' context-switching behavior, and we study how context-switching affects their productivity. We also survey developers to understand the reasons for and perceptions of multitasking. We find that the most common reason for multitasking is interrelationships and dependencies between projects. Notably, we find that the *rate of switching and breadth* (number of projects) of a developer's work matter. Developers who work on many projects have higher productivity if they focus on few projects per day. Developers that switch projects too much during the course of a day have lower productivity as they work on more projects overall. Despite these findings, developers' perceptions of the benefits of multitasking are varied.

CCS Concepts

•Information systems → Data analytics; •Human-centered computing → Empirical studies in collaborative and social computing; •Software and its engineering → Open source model;

Keywords

Multitasking; GitHub; productivity

Permission to make digital or hard copies of all or part of this work for personal classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICSE '16, May 14 - 22, 2016, Austin, TX, USA

© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ISBN 978-1-4503-3900-1/16/05...\$15.00

DOI: <http://dx.doi.org/10.1145/2884781.2884875>

1. INTRODUCTION

Multitasking is a staple of high performing professionals, programmers included. It is the ability to stop working on a task, switch to another, and return eventually to the first one, as needed or as scheduled. The goal is to optimize human resource allocation, while reprioritizing tasks dynamically [17]. When done well, or at least in a disciplined way, multitasking can yield dividends [1, 4]. Clearly, if a task in the queue has a higher priority than the current one, switching them can improve performance. For example, programmers encounter this when starting a new coding task, while their previous, urgently needed fix undergoes testing. If testing uncovers bugs, the urgent fix will require attention, and the new coding task will be put on the back-burner.

Multitasking comes at a cost though [9]. Humans, programmers included, have a certain, limited amount of *cognitive flexibility*, the mental ability to switch from thinking about one concept to thinking about another. Limitations apply to both the number of concepts we can juggle, as well as to the difficulty in switching between them. For example, as to the difficulty in switching between them. For example, coding simultaneously 7 simple functions is easier than coding 8 of them. Likewise, switching between two small matrix multiplication problems is more difficult than switching between two small integer additions. As we can imagine, reaching our innate limitations can result in decreased performance on all tasks and perhaps even diminished quality. It is unknown how far multitasking can be pushed safely, although some anecdotal evidence is available.

Software developers have long been pushing the limits on multitasking [27] because of the innate modularity of the development process and the independence of module processing (e.g., one can code while tests are being executed). In open-source software, developers also commonly contribute to multiple projects in the same time period, bridging different communities [20, 30]. With the advent of social coding tools like GitHub, this has intensified. It is not uncommon to find prolific developers contributing code to 5-10 GitHub projects in the same week. In fact, contributing to as many GitHub projects as possible is an accomplishment, valued by peers and employers alike [32].

There are various reasons why developers are more prolific on GitHub compared to other platforms. The features and usability provided by GitHub play a big role [34]. It is one example of how novel technology benefits programmers, and it empowers them; GitHub's platform is responsible for the inception of many projects, that otherwise wouldn't have existed [34]. And the payoffs are also substantial [34]. With

What your talk is for



Your paper
= the beef



Your talk
= the beef advertisement

What your talk is for

To give your audience an intuitive feel for your idea

To make them foam at the mouth with eagerness to read your paper

To engage, excite, provoke them

To make them glad they came
paper = the beef



Your talk
= the beef advertisement

What your talk is *not* for

To impress your audience
with your brainpower

To tell them everything you
know about your topic

To present all the technical
details

Your paper
= the beef



Your talk
= the beef advertisement

A photograph of two men, likely of Asian descent, looking down at a newspaper. The man on the left is wearing glasses and has white hair. The man on the right is wearing a suit and tie. They appear to be in a public setting, possibly a train or bus, as suggested by the blurred background.

**Don't lose
your audience**

1) Right structure

2) Adding stories



“Successful presenting is constantly answering the questions the audience has in mind”

‘Graham Davies is a brilliantly funny speaker who knows how to inspire and enthuse anyone who sees presenting as a bore, a burden or a source of terror’
Nick Robinson, Political Editor, BBC

The Presentation Coach

Bare Knuckle Brilliance
For Every Presenter

Graham Davies



“Successful presenting is constantly answering the questions the audience has in mind”

So how do you get the questions in?

‘Graham Davies is a brilliantly funny speaker who knows how to inspire and enthuse anyone who sees presenting as a bore, a burden or a source of terror’
Nick Robinson, Political Editor, BBC

The Presentation Coach

Bare Knuckle Brilliance
For Every Presenter

Graham Davies

Stories



Stories

A photograph of a campfire scene at night. In the foreground, a person wearing a cowboy hat and vest sits on the left, holding sticks over the fire. To their right, two other people sit facing the fire. The background is dark with silhouettes of tall trees. A yellow rectangular box is overlaid on the bottom right of the image, containing the text.

Of an average presentation, how much
is remembered after 3 hours?

A photograph of a group of people gathered around a campfire at night. A man on the left is wearing a cowboy hat and holding sticks with marshmallows over the flames. A woman in a yellow tank top is seated to his right. In the background, other people are visible under a dark sky with some stars.

Stories

40%

Of an average presentation, how much
is remembered after 3 hours?

Stories

A photograph of a group of people gathered around a campfire at night. A man on the left, wearing a cowboy hat and holding sticks, is roasting marshmallows over the flames. To his right, a woman in a yellow tank top and a man in a white t-shirt are sitting and watching the fire. The background is dark with silhouettes of trees and a bright moon in the sky.

Of an average presentation, how much
is remembered after 3 **months**?

Stories

5%

A photograph of a group of people gathered around a campfire at night. A man on the left is holding a stick with marshmallows over the flames. Two women are seated to his right, looking towards the fire. The scene is lit by the warm glow of the fire, with dark silhouettes of trees in the background.

Of an average presentation, how much
is remembered after 3 months?

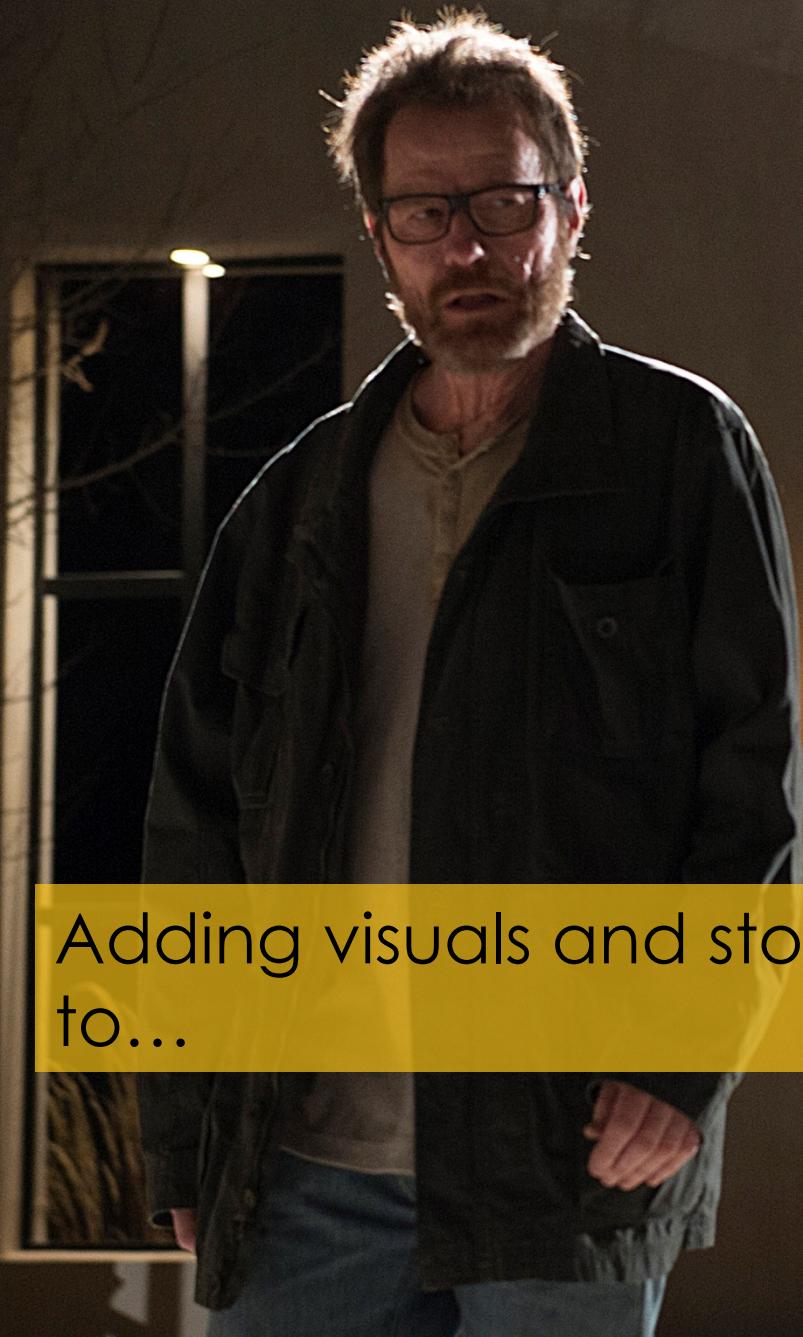




Adding visuals and storyline increases this
to...

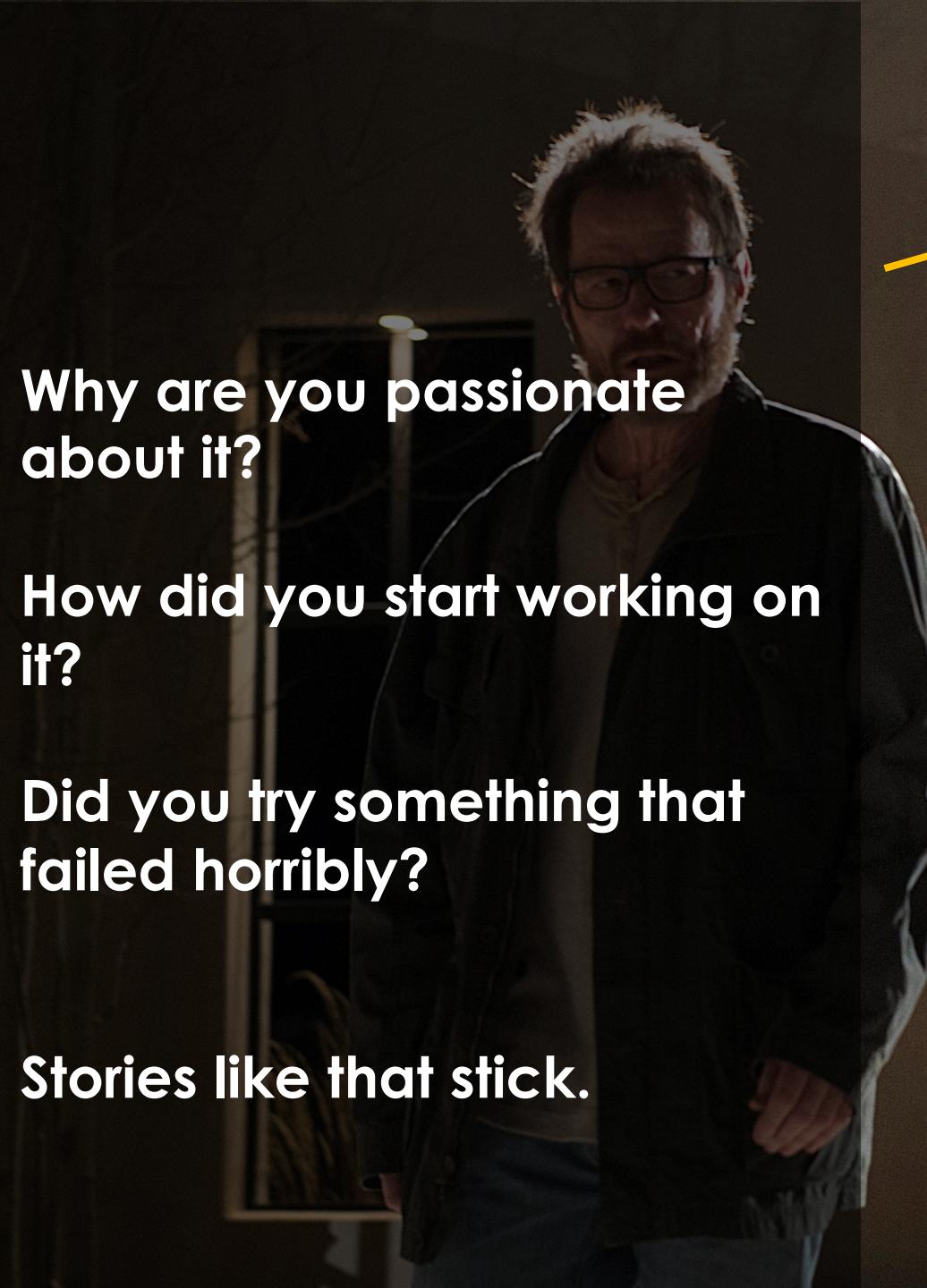
60%

Adding visuals and storyline increases this to...





Yes... but...
my topic...



Yes... but...
my topic...

Why are you passionate
about it?

How did you start working on
it?

Did you try something that
failed horribly?

Stories like that stick.

A good story



Why Some Ideas Survive and Others Die...

MADE to STICK

Chip Heath & Dan Heath

A good story

S Simple
U Unexpected
C Credible
C Concrete
E Emotional
S Stories



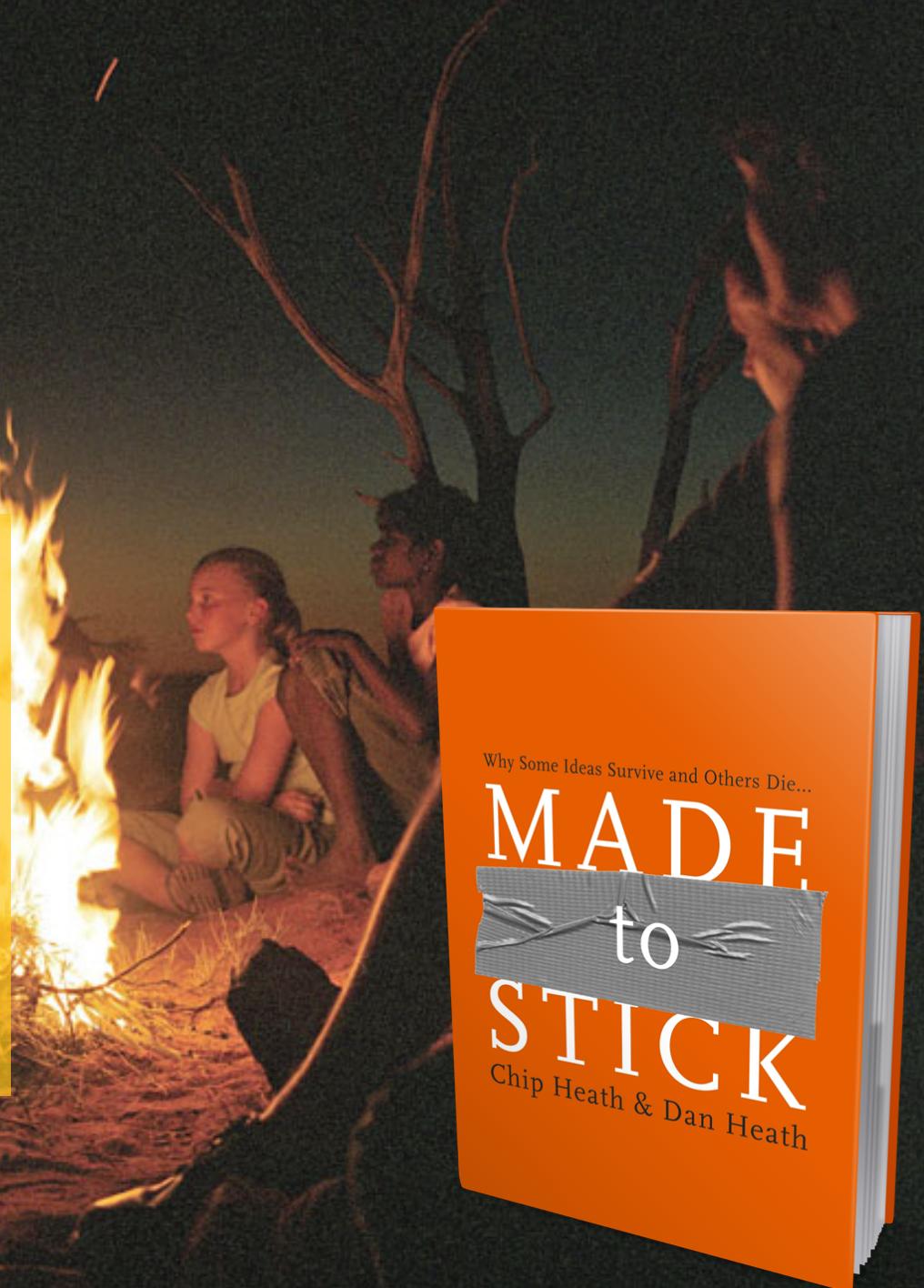
Why Some Ideas Survive and Others Die...

MADE
to
STICK

Chip Heath & Dan Heath

A good story

S Simple
U Unexpected
C Credible
C Concrete
E Emotional
S Stories



Examples stick

S Simple
U Unexpected
C Credible
C Concrete
E Emotional
S Stories

Use examples:

- To motivate the work
- To convey the basic intuition
- To illustrate The Idea in action
- To show extreme cases
- To highlight shortcomings



**Don't lose
your audience**

1) Right structure

2) Adding stories

3) Support

Don't lose your audience

Once you have the right structure and memorable stories, the last thing you can do is support your presentation in the right way.

Supporting is done with:

- Beautiful slides
- Good body language
- Clear use of your voice/speech

1) Right structure

2) Adding stories

3) Support

My preferences for slide design:

- Consistent style
- Cohesive: one thought per slide
- Little text: audience reading slides stops listening to you
- Title is informative and does not repeat
- Corporate template depending on audience

Awesome slides

THE END

Questions???

Remember, this slide will most likely be on the screen the longest. So:

- Acknowledge sponsors
- Add your contact details
- Summarize your results to **encourage questions** (slide minis)
- Questions are not a problem. Questions are a **golden golden golden** opportunity to connect with your audience

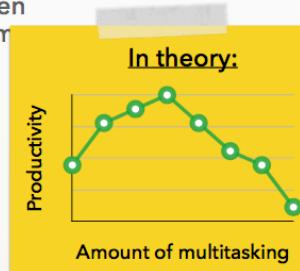
Theory: How does multitasking affect performance?

PROS

► **Fill downtime**

Switch focus between projects to utilize time more efficiently

(Adler and Benbanun-Fich, 2012)



► **Cross-fertilisation**

Easier to work on other projects if knowledge is transferrable

(Lindbeck and Snower, 2000)

CONS

► **Cognitive switching cost**

Depends on interruption duration, complexity, moment

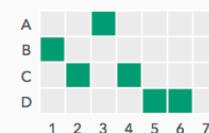
Altmann and Trafton, 2002
Borst, Taatgen, van Rijn, 2015

"Project overload"

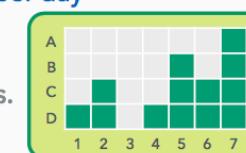
Mental congestion when too much multitasking
(Zika-Viktorsson, Sundstrom, Engwall, 2006)

Multitaskers do more; scheduling matters

Projects per day

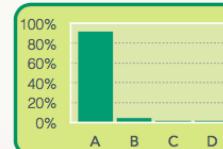


VS.

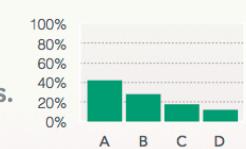


More within-day multitasking

Weekly focus

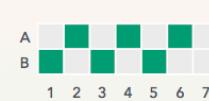


VS.

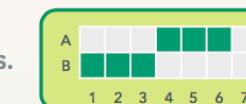


Higher focus
More repetitive day-to-day work

Day-to-day focus (repeatability)



VS.



Interaction effects:
No scheduling is productive over 5 projects/week

Body language

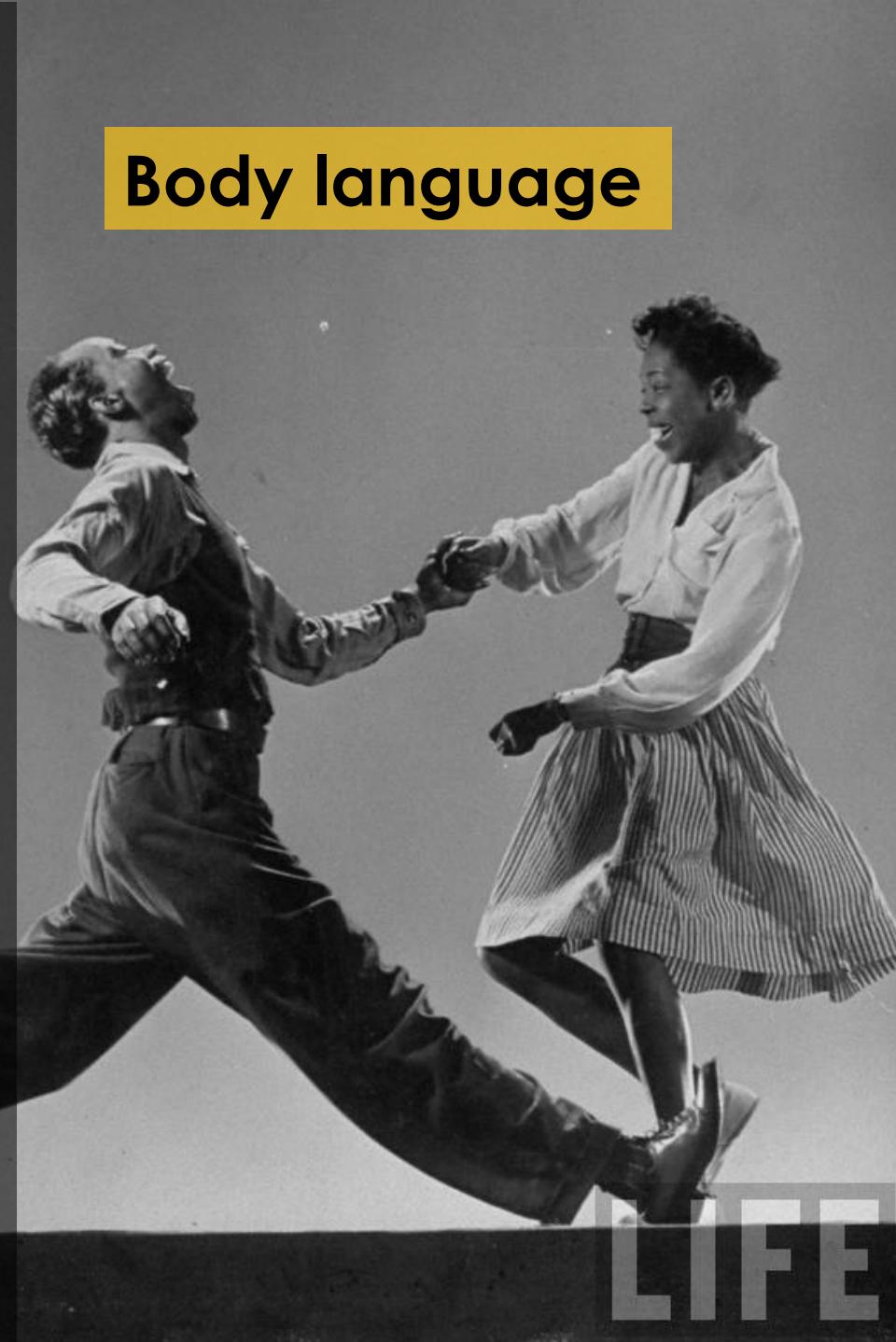
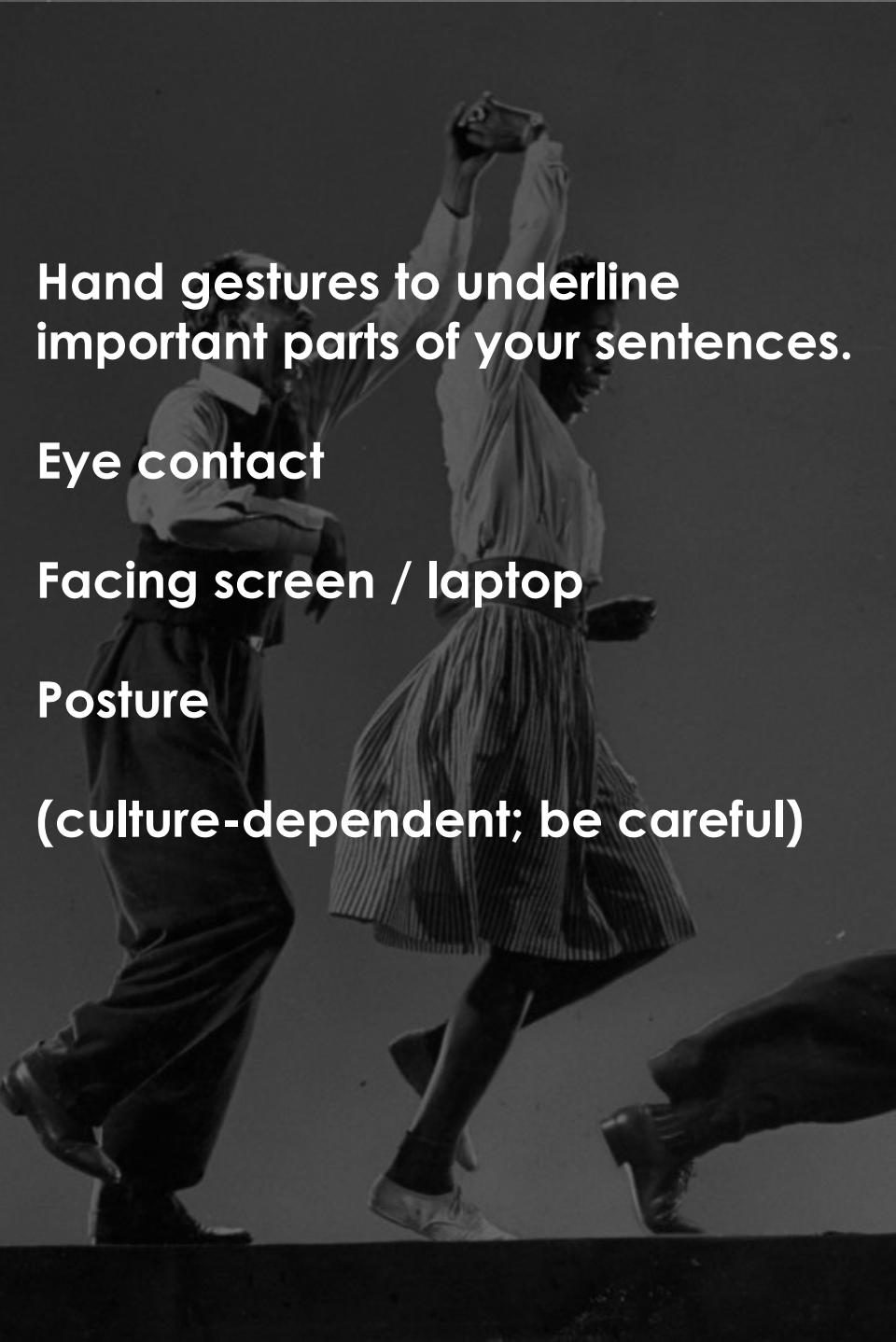
Hand gestures to underline important parts of your sentences.

Eye contact

Facing screen / laptop

Posture

(culture-dependent; be careful)





The final tool for supporting your talk is your voice.

- **Enthusiasm** makes people dramatically more receptive.
- **Pauses help**
 - Pause at least before and after important sentences.
 - Ironically, the best way to get a room's attention is to shut up!



**Don't lose
your audience**

1) Right structure

2) Adding stories

3) Support



**Don't lose
your audience**

1) Right structure

2) Adding stories

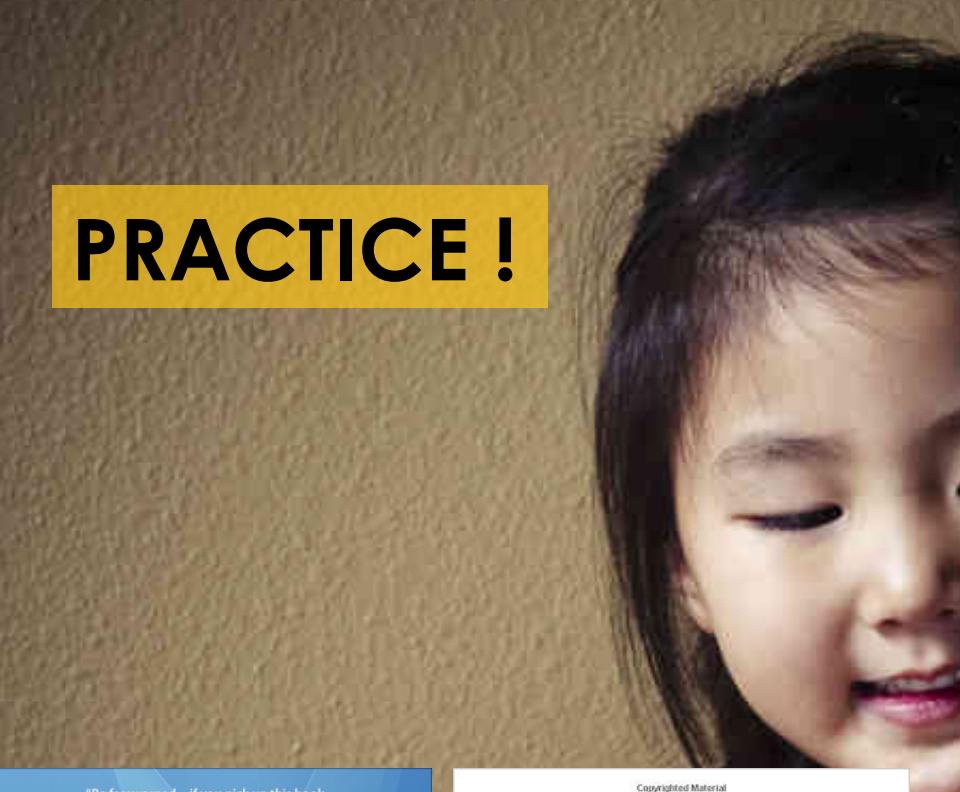
3) Support

One more thing...

PRACTICE !

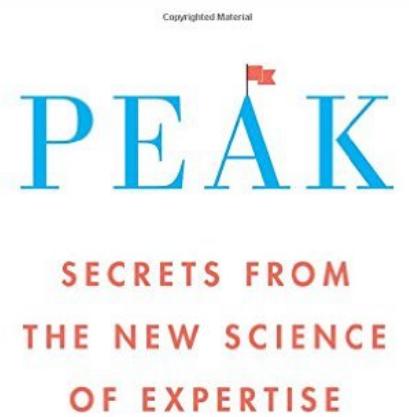
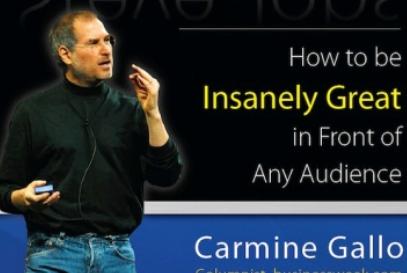


PRACTICE !



"Be forewarned—if you pick up this book,
your presentations will never be the same again."
—Martin Lindstrom, bestselling author of *Buyology*

The Presentation Secrets of Steve Jobs



Anders Ericsson
and Robert Pool

No one is born a violin player or a marathon runner. Everyone needs practice.

Even Steve Jobs practised for every talk.

Pittsburgh has:

- Toast masters club
- Improv comedy club
- ...

If you remember nothing else, remember this:



Your paper
= the beef



Your talk
= the beef advertisement



Slides by:
[Felienne Hermans](#)
[Simon Peyton Jones](#)
liberally edited