How to Write a Successful Conference Proposal

Steven Syrek (@sjsyrek)

github.com/sjsyrek/presentations/tree/master/how-to-write-a-successful-conference-proposal

Know your audience

Pick something

Be original

Get started early

Brainstorm first

Start writing before you know what you want to say

Iterate until you do know what you want to say

Grow and prune

Focus on what you can reasonably achieve

Aim for being sharp and succinct

Avoid being terse and vague

Avoid being voluminous

Proofread your work

Get feedback

Sleep on it

Don't do it at the last minute

Don't be cute, clever, or funny

Don't show off

Don't assume your readers will be familiar with your topic

Don't assume anyone can read your mind

Don't use sophisticated language or excessive amounts of jargon

Don't be evasive, diffident, or obfuscatory

Don't use passive voice

Don't be casual

Don't be negative

Don't be afraid to ask for help

Be confident

Remember your audience

Have you ever wanted to implement a programming language from scratch? Doing so can be a rewarding and educational experience, and it's probably much easier than you think. We will guide you through the hoops of creating a small, but neat, programming language. By the end of the workshop you will be the proud author of a programming language, and you will definitely have a better understanding of how programming languages work.

We all enjoy meeting up at conferences, but what are you supposed to do the rest of the time? How can you best enhance your skills before the books, blogs, and docs come out? We'll discuss how to learn from code in the wild, focusing on the newest language patterns used in open source JavaScript libraries.

A hands-on introduction to Idris, a general-purpose functional programming language with dependent types.

This session is a hands-on workshop that covers the development of web applications using Haskell with the web framework Yesod. Attendees will write a fully functional blogging system, including post authoring, user login and authorization, JSON API access, and commenting, with all site data stored persistently in a database. The workshop will teach the specifics of creating a web server in Yesod in addition to explaining the abstractions and technologies involved in the process. In doing so, common tasks such as safely handling user supplied input and dealing with database interaction will be covered in a purely functional language. In addition, the mechanisms through which strong static typing and functional programming can improve the safety and readability of programs will be examined, using Yesod as a case study.

This session will introduce the concept of the Monad Transformer, explain why it is useful and what its drawbacks are, and finally compare its representation in various functional programming languages.

In this hands-on, coding workshop, participants will develop an intuition for functional programming fundamentals by implementing a basic, untyped lambda calculus right in their browsers. Using hardly anything more than ES6 arrow functions and a few translation utilities, we will reinvent the entire wheel of Turing-complete computation by exploring the nature of functions, the utility of abstraction, and the basis for FP concepts such as purity, recursion, referential transparency, currying, composition, and more. By concentrating on specific patterns with practical examples, we will explore this classic foundation of computer science in the safe and familiar environment of the JavaScript console. As we will be focusing on the basics and eschewing the added complexities introduced by type systems, this workshop is most suitable for total beginners to programming and experienced developers who are newcomers to FP. Anyone seeking the opportunity to take a closer look at the core concepts underlying FP, however, is likely to get something out of attending. Participants can expect to learn what it means to treat functions as values and how to build complex programs out of the simplest building blocks—the necessary prerequisites for the ultimate goal of adopting, loving, and living the functional life.

Programming is a relatively new pursuit, but it is one that depends on ancient, evolved abilities such as language processing, abstract reasoning, and pattern recognition. These inherent skills are performed by various brain regions specialized for particular functions, leading to the question: what does the brain of a programmer look like while coding? This talk proposes to cover the current literature investigating what our ancient brains do when we interact with computers.

As functional programming has gained traction in the software industry, more and more companies are discovering the advantages of development and debugging it provides. Software that is easier to develop and debug is, by definition, more efficient and elegant, and functional concepts are gradually becoming mainstream. The way we compose software has a huge impact on our processes from day to day. As the various functional frameworks evolve over time, we gain the benefits of constant improvement in our tools as well as being guided toward the most effective design decisions. Our industry has finally reached a major turning point, and static typing in addition to functional features are gradually being incorporated into the newest, most popular programming languages. In the near future, functional programming will clearly become the paradigm most developers turn to in order to boost their productivity and effciency. I plan to discuss the benefits of functional programming for the software industry today as well as strategies for incorporating functional programming concepts into mainstream practices and the languages developers are using to build software. We will also use a case study that shows how functional programming techniques motivated a major project at my company and how it created productivity gains for our team and value for the business, which would not have been possible using a different approach.

In this talk, I will correct the mistakes and false beliefs that many programmers have about algebraic data types and the algebraic qualities of functional programming languages. I will explain the underlying mathematical concepts in order to help them better understand their code and how it really relates to this mathematical domain.

GraphQL dared to ask, “How can we make REST better for interactive web and mobile applications?” Interestingly, its solution involved the introduction of a type language for shared API understanding. The speaker is currently writing a library in Scala and Scala.js which embeds this kind of type language into Scala types themselves, so that we can get compile-time guarantees of API compliance similar to what has been successfully pioneered by the Servant REST API library in Haskell. This talk discusses what we can learn from GraphQL, how types can help define APIs even when they're distributed, and the API of such a library in Scala.

— Know your audience
— Pick something
— Be original
— Get started early
— Brainstorm first
— Start writing before you know what you want to say
— Iterate until you do know what you want to say
— Grow and prune
— Focus on what you can reasonably achieve
— Aim for being sharp and succinct
— Avoid being terse and vague
— Avoid being voluminous
— Proofread your work
— Get feedback
— Sleep on it
— Don't do it at the last minute
— Don't be cute, clever, or funny
— Don't show off
— Don't assume your readers will be familiar with your topic
— Don't assume anyone can read your mind
— Don't use sophisticated language or excessive amounts of jargon
— Don't be evasive, diffident, or obfuscatory
— Don't use passive voice
— Don't be casual
— Don't be negative
— Be confident
— Remember your audience