

# The future is mostly static

Or how we are reinventing the web again

---

Juho Vepsäläinen

# What to expect

1. Introduction
2. Content Management Systems (CMSs)
3. Static Site Generation (SSG)
4. CMS and SSG compared
5. Current trends
6. Research

# Introduction

---

## Key points from the history of the web

- 1985 - First Content Management System (CMS) called FileNet is invented

## Key points from the history of the web

- 1985 - First Content Management System (CMS) called FileNet is invented
- 1989 - First HTML browser comes available

## Key points from the history of the web

- 1985 - First Content Management System (CMS) called FileNet is invented
- 1989 - First HTML browser comes available
- 1992 - Tim Berners-Lee invents the World Wide Web (www) [4]

## Key points from the history of the web

- 1985 - First Content Management System (CMS) called FileNet is invented
- 1989 - First HTML browser comes available
- 1992 - Tim Berners-Lee invents the World Wide Web (www) [4]
- ca. 1995 - First graphical editors (MS Frontpage etc.) emerge for creation of websites

# Key points from the history of the web

- 1985 - First Content Management System (CMS) called FileNet is invented
- 1989 - First HTML browser comes available
- 1992 - Tim Berners-Lee invents the World Wide Web (www) [4]
- ca. 1995 - First graphical editors (MS Frontpage etc.) emerge for creation of websites
- 1995 - JavaScript is developed by Brendan Eich



# Key points from the history of the web

- 1985 - First Content Management System (CMS) called FileNet is invented
- 1989 - First HTML browser comes available
- 1992 - Tim Berners-Lee invents the World Wide Web (www) [4]
- ca. 1995 - First graphical editors (MS Frontpage etc.) emerge for creation of websites
- 1995 - JavaScript is developed by Brendan Eich
- 2002 - The first Single Page Applications (SPAs) emerge

# Key points from the history of the web

- 1985 - First Content Management System (CMS) called FileNet is invented
- 1989 - First HTML browser comes available
- 1992 - Tim Berners-Lee invents the World Wide Web (www) [4]
- ca. 1995 - First graphical editors (MS Frontpage etc.) emerge for creation of websites
- 1995 - JavaScript is developed by Brendan Eich
- 2002 - The first Single Page Applications (SPAs) emerge
- 2015 - [Jamstack is specified by Matt Biilmann \[9\]](#)

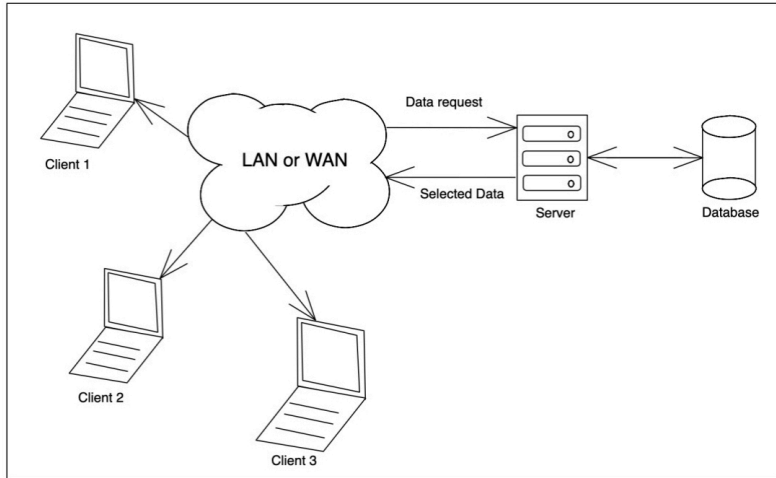
# Key points from the history of the web

- 1985 - First Content Management System (CMS) called FileNet is invented
- 1989 - First HTML browser comes available
- 1992 - Tim Berners-Lee invents the World Wide Web (www) [4]
- ca. 1995 - First graphical editors (MS Frontpage etc.) emerge for creation of websites
- 1995 - JavaScript is developed by Brendan Eich
- 2002 - The first Single Page Applications (SPAs) emerge
- 2015 - Jamstack is specified by Matt Biilmann [9]
- 2019 - Islands architecture is formalized [16]

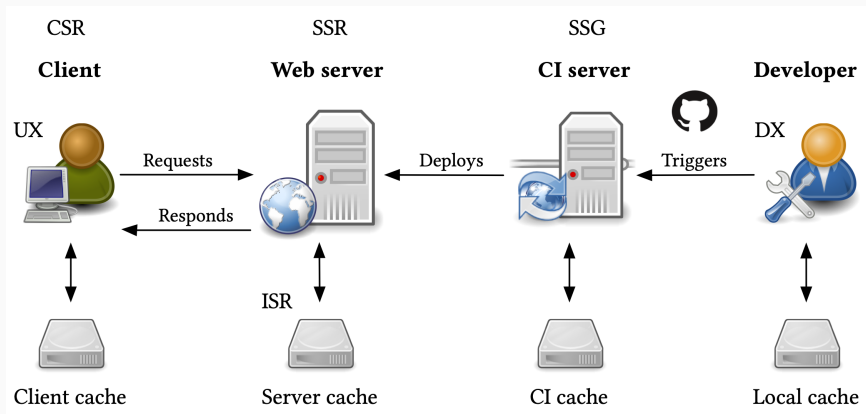
# Key points from the history of the web

- 1985 - First Content Management System (CMS) called FileNet is invented
- 1989 - First HTML browser comes available
- 1992 - Tim Berners-Lee invents the World Wide Web (www) [4]
- ca. 1995 - First graphical editors (MS Frontpage etc.) emerge for creation of websites
- 1995 - JavaScript is developed by Brendan Eich
- 2002 - The first Single Page Applications (SPAs) emerge
- 2015 - Jamstack is specified by Matt Biilmann [9]
- 2019 - Islands architecture is formalized [16]
- 2021 - The idea of Transitional Web Applications (TWAs) is proposed [8]

# Clients and a server [9]



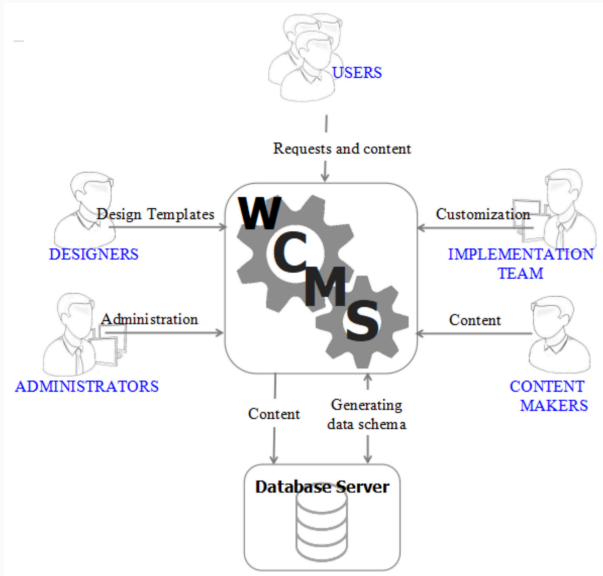
# Client, server, developer



# Content Management Systems (CMSs)

---

# Roles in CMSs [18]





# What is a CMS? [3]

## Content Management Products

CMS

Authoring  
Acquisition  
Conversion  
Aggregation  
Collection  
Services

Collection

Repository  
Administration  
Workflow

Management

Template  
System  
Publication  
Services  
Website Support  
Other Media  
Support

Publishing

# Pros and cons

## Pros

- Enables collaboration across disciplines

## Cons

# Pros and cons

## Pros

- Enables collaboration across disciplines
- Allows developers to save time by using extensions

## Cons

# Pros and cons

## Pros

- Enables collaboration across disciplines
- Allows developers to save time by using extensions
- Many major players exist in the ecosystem

## Cons

# Pros and cons

## Pros

- Enables collaboration across disciplines
- Allows developers to save time by using extensions
- Many major players exist in the ecosystem

## Cons

- Often requires a server (something to maintain)

# Pros and cons

## Pros

- Enables collaboration across disciplines
- Allows developers to save time by using extensions
- Many major players exist in the ecosystem

## Cons

- Often requires a server (something to maintain)
- Not trivial to understand

# Pros and cons

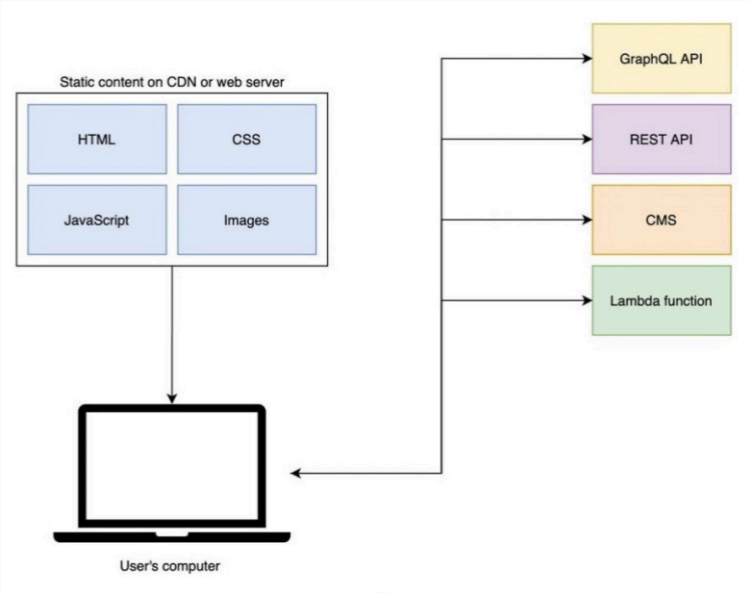
## Pros

- Enables collaboration across disciplines
- Allows developers to save time by using extensions
- Many major players exist in the ecosystem

## Cons

- Often requires a server (something to maintain)
- Not trivial to understand
- Comes sometimes with unnecessary complexity

# Architecture of a headless CMS [2, 12]





# Static Site Generation (SSG)

---

# What is SSG? [11]

SSG is a system consisting of the following:

1. A templating language for website layout and theming.

# What is SSG? [11]

SSG is a system consisting of the following:

1. A templating language for website layout and theming.
2. A markup language for content creation (e.g., Markdown).

# What is SSG? [11]

SSG is a system consisting of the following:

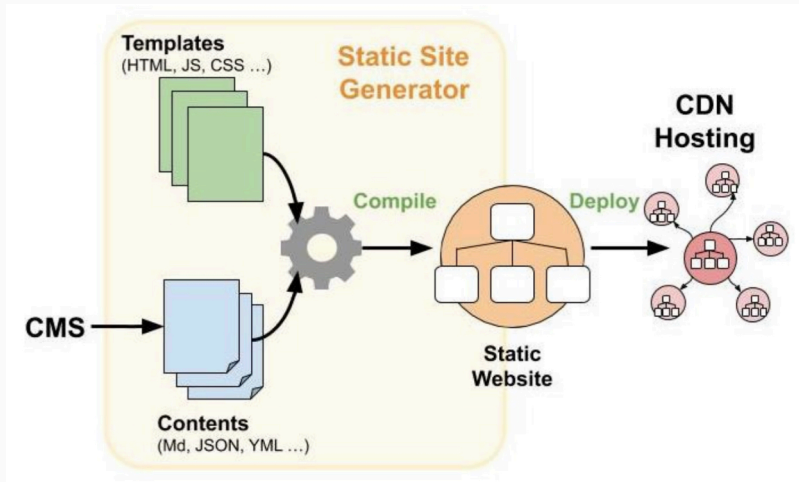
1. A templating language for website layout and theming.
2. A markup language for content creation (e.g., Markdown).
3. A local development server to preview and test the site before building.

# What is SSG? [11]

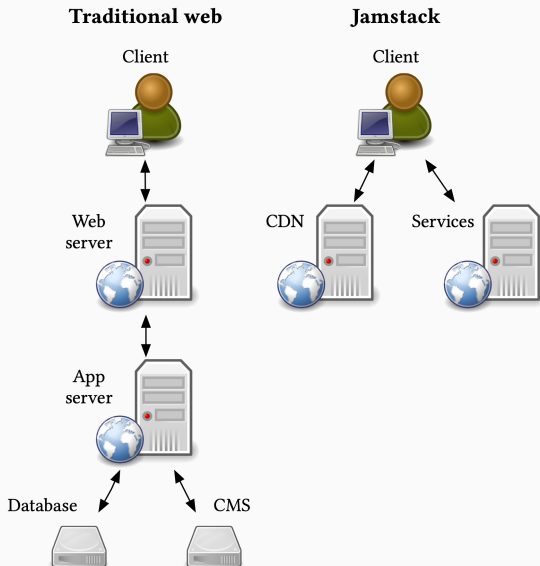
SSG is a system consisting of the following:

1. A templating language for website layout and theming.
2. A markup language for content creation (e.g., Markdown).
3. A local development server to preview and test the site before building.
4. A compile process that builds the final site files into HTML, CSS, and JavaScript [5]

# Building blocks of a Jamstack website [17]



# Traditional web compared to Jamstack [13]



# Pros and cons

## Pros

- Fast page load time [14, 5]

## Cons



# Pros and cons

## Pros

- Fast page load time [14, 5]
- Scalable and resilient [14]

## Cons

# Pros and cons

## Pros

- Fast page load time [14, 5]
- Scalable and resilient [14]
- Availability and security [14, 5]

## Cons

# Pros and cons

## Pros

- Fast page load time [14, 5]
- Scalable and resilient [14]
- Availability and security [14, 5]
- [Automatic versioning](#) [5]

## Cons

# Pros and cons

## Pros

- Fast page load time [14, 5]
- Scalable and resilient [14]
- Availability and security [14, 5]
- Automatic versioning [5]
- [Ease of hosting \[11\]](#)

## Cons

# Pros and cons

## Pros

- Fast page load time [14, 5]
- Scalable and resilient [14]
- Availability and security [14, 5]
- Automatic versioning [5]
- Ease of hosting [11]

## Cons

- Tooling is highly developer oriented

# Pros and cons

## Pros

- Fast page load time [14, 5]
- Scalable and resilient [14]
- Availability and security [14, 5]
- Automatic versioning [5]
- Ease of hosting [11]

## Cons

- Tooling is highly developer oriented
- Possibly high recompilation cost

# Pros and cons

## Pros

- Fast page load time [14, 5]
- Scalable and resilient [14]
- Availability and security [14, 5]
- Automatic versioning [5]
- Ease of hosting [11]

## Cons

- Tooling is highly developer oriented
- Possibly high recompilation cost
- **Static by definition**  
(difficulties in dynamic use cases)

## CMS and SSG compared

---



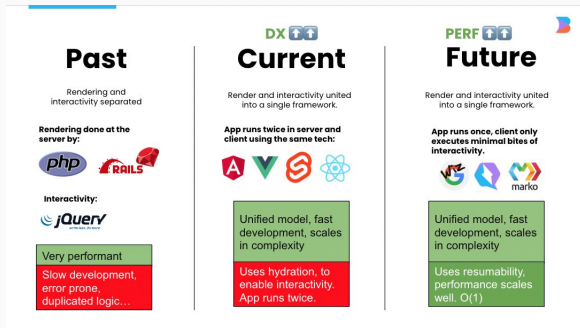
# CMS vs. SSG

Aspect	CMS	SSG
Focus	Collaboration across disciplines	Developer productivity
Security	Needs constant maintenance	Only static file server to secure
Technical complexity	Complex by definition	Potentially very simple
Dynamic use	Supported out of the box	Static by definition (third parties)

## Current trends

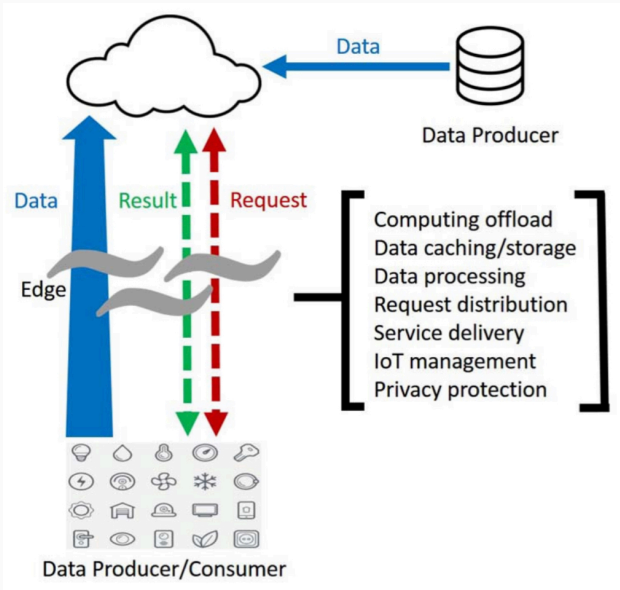
---

# Past, present, future



Source: Misko Hevery: WebApps at Scale

# Edge computing [15]



## Transitional web applications (TWAs) [8]

- In October 2021, Rich Harris proposed the idea of TWAs [8]

# Transitional web applications (TWAs) [8]

- In October 2021, Rich Harris proposed the idea of TWAs [8]
- By his definition, TWAs mix ideas from SPAs and the traditional web

# Transitional web applications (TWAs) [8]

- In October 2021, Rich Harris proposed the idea of TWAs [8]
- By his definition, TWAs mix ideas from SPAs and the traditional web
- TWAs utilize SSR for fast initial loading times

# Transitional web applications (TWAs) [8]

- In October 2021, Rich Harris proposed the idea of TWAs [8]
- By his definition, TWAs mix ideas from SPAs and the traditional web
- TWAs utilize SSR for fast initial loading times
- TWAs are resilient as they work without JavaScript by default



# Transitional web applications (TWAs) [8]

- In October 2021, Rich Harris proposed the idea of TWAs [8]
- By his definition, TWAs mix ideas from SPAs and the traditional web
- TWAs utilize SSR for fast initial loading times
- TWAs are resilient as they work without JavaScript by default
- TWAs provide consistent experience and accessibility as a built-in feature

## Progressive enhancement in a nutshell (2008) [7]

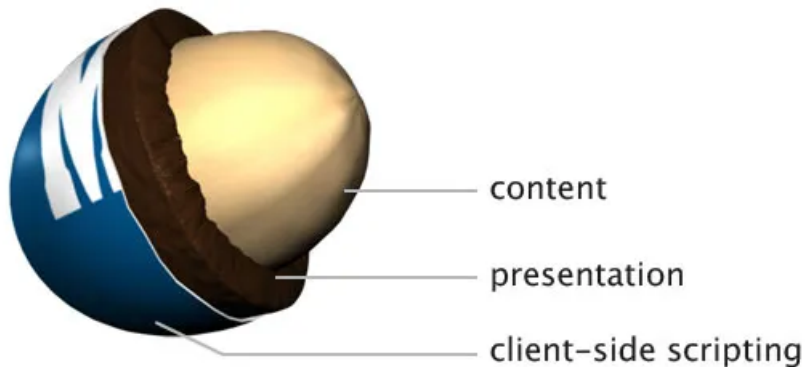


Illustration by Dave Stewart

## Disappearing frameworks [6]

- According to Ryan Carniato [6], disappearing frameworks come with almost a zero cost and disappear from an application

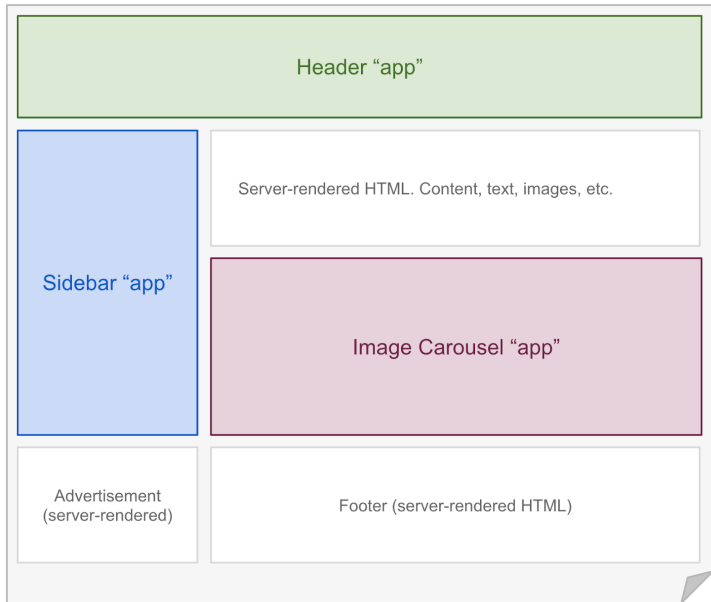
## Disappearing frameworks [6]

- According to Ryan Carniato [6], disappearing frameworks come with almost a zero cost and disappear from an application
- The starting point forms a contrast to the current breed of frameworks that load upfront and rely on expensive hydration for SSR

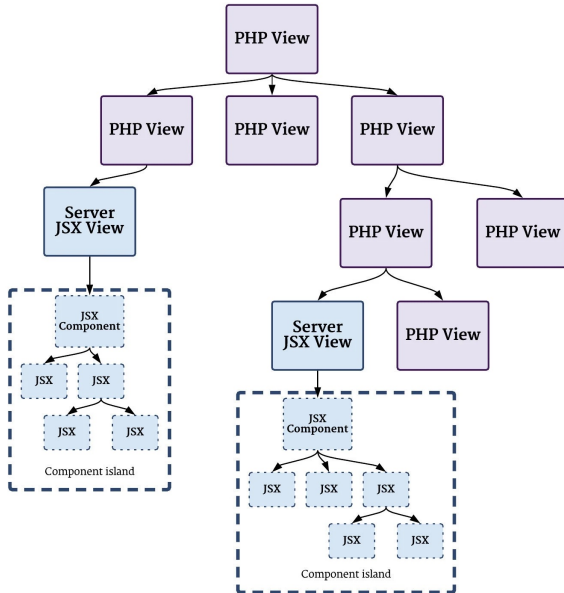
## Disappearing frameworks [6]

- According to Ryan Carniato [6], disappearing frameworks come with almost a zero cost and disappear from an application
- The starting point forms a contrast to the current breed of frameworks that load upfront and rely on expensive hydration for SSR
- Changing the fundamental viewpoint allows for new architectures to emerge and it's consistent with the idea of TWAs

# Islands architecture [10]



# Islands architecture implemented by Etsy [1]



- Astro - Lots of integrations, interesting model for mixing interactivity with content



- Astro - Lots of integrations, interesting model for mixing interactivity with content
- îles - Powered by Vite, supports many frameworks (Vue, React, Preact, etc.) for islands

- Astro - Lots of integrations, interesting model for mixing interactivity with content
- îles - Powered by Vite, supports many frameworks (Vue, React, Preact, etc.) for islands
- Capri - Powered by Vite, live CMS integration, early release

- SolidJS - React done "right"

## Upcoming technologies

- SolidJS - React done "right"
- Qwik - No hydration, automatic lazy loading, and more

# Upcoming technologies

- SolidJS - React done "right"
- Qwik - No hydration, automatic lazy loading, and more
- [Nano JSX - 1 kB JSX library](#)

# Upcoming technologies

- SolidJS - React done "right"
- Qwik - No hydration, automatic lazy loading, and more
- Nano JSX - 1 kB JSX library
- Tina - Open source editor for Next.js sites

# Upcoming technologies

- SolidJS - React done "right"
- Qwik - No hydration, automatic lazy loading, and more
- Nano JSX - 1 kB JSX library
- Tina - Open source editor for Next.js sites
- Contentlayer - Type-safe JSON APIs on top of Markdown and other formats

# Upcoming technologies

- SolidJS - React done "right"
- Qwik - No hydration, automatic lazy loading, and more
- Nano JSX - 1 kB JSX library
- Tina - Open source editor for Next.js sites
- Contentlayer - Type-safe JSON APIs on top of Markdown and other formats
- **Portable Text - Abstraction of rich text enabling creation of editors (powers Sanity)**



# Upcoming technologies

- SolidJS - React done "right"
- Qwik - No hydration, automatic lazy loading, and more
- Nano JSX - 1 kB JSX library
- Tina - Open source editor for Next.js sites
- Contentlayer - Type-safe JSON APIs on top of Markdown and other formats
- Portable Text - Abstraction of rich text enabling creation of editors (powers Sanity)
- Fresh - Web framework with zero runtime overhead, islands, no build step, no configuration, oriented around (P)React

# Research

---

1. What are the main benefits and limitations of CMSs and SSGs?

# Research questions

1. What are the main benefits and limitations of CMSs and SSGs?
2. What are the means in which SSGs could address their limitations compared to CMSs?

# Research questions

1. What are the main benefits and limitations of CMSs and SSGs?
2. What are the means in which SSGs could address their limitations compared to CMSs?
3. What could the future of SSGs look like?

- By nature, design science fits the problem well

- By nature, design science fits the problem well
- Besides literature review, the idea is to:

- By nature, design science fits the problem well
- Besides literature review, the idea is to:
  1. Interview practitioners and tool authors to understand how they view the field



- By nature, design science fits the problem well
- Besides literature review, the idea is to:
  1. Interview practitioners and tool authors to understand how they view the field
  2. Construct a model of what SSGs with a high amount of dynamism could look like

- By nature, design science fits the problem well
- Besides literature review, the idea is to:
  1. Interview practitioners and tool authors to understand how they view the field
  2. Construct a model of what SSGs with a high amount of dynamism could look like
  3. Implement a tool to try out the ideas in practice (Gustwind, in progress)

1. Increased understanding of the current state of the art

# Contribution to web engineering

1. Increased understanding of the current state of the art
2. Understanding of what SSG could look like when combined with ideas and constraints from the CMS world





# Contribution to web engineering

1. Increased understanding of the current state of the art
2. Understanding of what SSG could look like when combined with ideas and constraints from the CMS world
3. Creation of tooling for the next generation of web developers to bridge the gap

Thank you!

Time for your questions

# References i

-  Etsy engineering: Mobius: Adopting jsx while prioritizing user experience, 2021.
-  J. Attardi.  
**Introduction to netlify cms.**  
In *Using Gatsby and Netlify CMS*, pages 1–12. Springer, 2020.
-  C. Benevolo and S. Negri.  
**Evaluation of content management systems (cms): a supply analysis.**  
*Electronic Journal of Information Systems Evaluation*, 10(1):pp9–22, 2007.
-  T. Berners-Lee, R. Cailliau, J.-F. Groff, and B. Pollermann.  
**World-wide web: the information universe.**  
*Internet Research*, 1992.



## References ii



R. Camden and B. Rinaldi.

*Working with Static Sites: Bringing the Power of Simplicity to Modern Sites.*

"O'Reilly Media, Inc.", 2017.



R. Carniato.

**Understanding transitional javascript apps, Nov 2021.**



A. Gustafson, L. Overkamp, P. Brosset, S. V. Prater, M. Wills, and E. PenzeyMoog.

**Understanding progressive enhancement, Oct 2008.**



R. Harris.

**Have single-page apps ruined the web? | transitional apps with rich harris, nytimes, Oct 2021.**

## References iii



S. Kumar.

**A review on client-server based applications and research opportunity.**

*International Journal of Recent Scientific Research*,  
10(7):33857–3386, 2019.



J. Miller.

**Islands architecture, 2020.**



K. Newson.

**Tools and workflows for collaborating on static website projects.**

*Code4Lib Journal*, (38), 2017.



E. O. Obaseki et al.

***Micro-frontends for Web Content Management Systems.***

PhD thesis, Covenant University, 2021.

-  S. Peltonen, L. Mezzalana, and D. Taibi.  
**Motivations, benefits, and issues for adopting micro-frontends: a multivocal literature review.**  
*Information and Software Technology*, 136:106571, 2021.
-  H. Petersen.  
**From static and dynamic websites to static site generators.**  
*university of TARTU, Institute of Computer Science*, 2016.
-  W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu.  
**Edge computing: Vision and challenges.**  
*IEEE internet of things journal*, 3(5):637–646, 2016.
-  T. P. team.  
**Islands architecture**, Oct 2021.



J. Väänänen.

**Jamstack–dynaamisesti toimiva staattinen verkkosivusto.**  
2019.



A. Yermolenko and Y. Golchevskiy.

**Developing web content management systems–from the past to the future.**

In *SHS Web of Conferences*, volume 110. EDP Sciences, 2021.