Gatsby

Statische Webseiten, super optimiert

Dipl.-Ing. Franz Knipp

Linuxwochen Eisenstadt, 21. April 2018

Linuxwochen – seit ???

2009 Openoffice.org

2010 Inkscape

2011 Sweet Home 3D

2012 jQuery – JavaScript 2.0

2014 Einbetten von Python in C/C++ Programmen

2015 Wie ich 10- bis 14-Jährigen das Programmieren beibringe

2016 Meteor – die Open Source Javascript App Plattform

2017 React Native: JavaScript am Smartphone auf der Überholspur

Und heute?

Von allem ein bisschen was

JavaScript, React, Node, ein bisschen SVG

OpenOffice :-)

Was ist Gatsby?

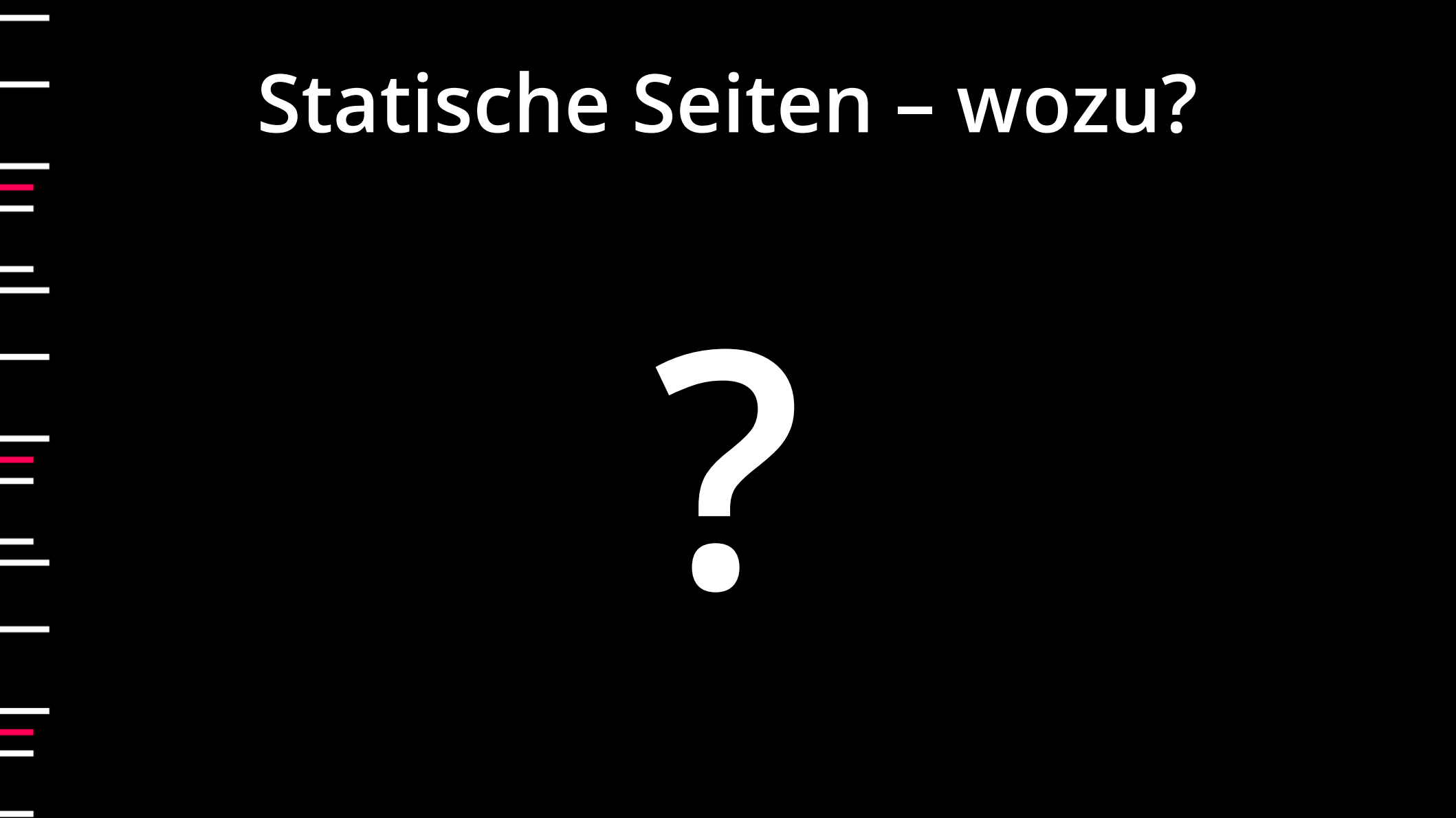
- Eine Romanfigur von F. Scott Fitzgerald

Was ist Gatsby?

- ~~Eine Romanfigur von F. Scott Fitzgerald~~
- Ein Generator für statische Web-Seiten
- Mit ganz vielen neuen Technologien
- https://www.gatsbyjs.org/
- Seit 2015, MIT-Lizenz

Gatsby

Statische Seiten – wozu?



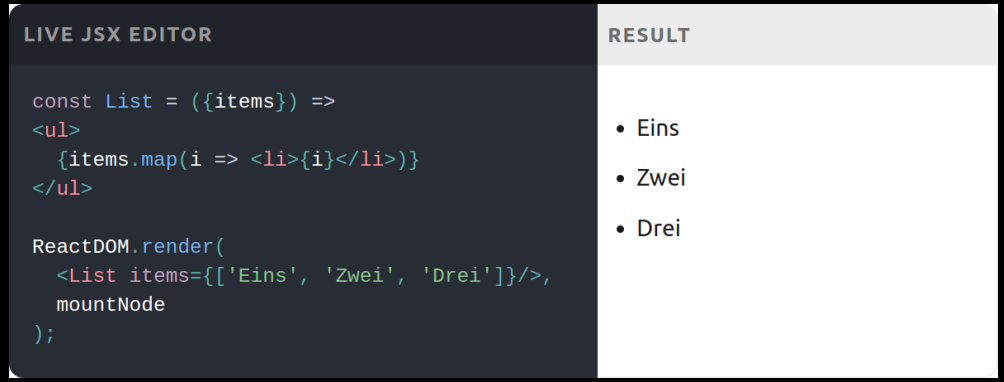
Static Site Generator

Das ist nicht neu!

React

- Komponentenbasiertes Webframework
- HTML in JavaScript (JSX)
- Virtual DOM

- Facebook, 2013
- MIT-Lizenz
- > 93.000 Github stars
- https://reactjs.org/



GraphQL

- Abfragesprache
- Alternative zu REST
- Client bestimmt Daten
- Facebook, 2015
- Mehrere OpenSource-Implementierungen: Apollo, Relay
- http://graphql.org/

```
query HumanNameAndFriends {
  human(id:1000) {
    name
    friends {
      name
    }
  }
}
```

```
{
  "data": {
    "human": {
      "name": "Luke Skywalker",
      "friends": [
        {
          "name": "Han Solo"
        },
        {
          "name": "Leia Organa"
        },
        {
          "name": "C-3PO"
        },
        {
          "name": "R2-D2"
        }
      ]
```

Webpack

- Module Bundler
  - Erzeugt statische Assets (JS, CSS, HTML) aufgrund von Modul-Abhängigkeiten
- Super-tolle Developer Experience dank Hot Module Reload
- Muss zum Glück nicht selbst konfiguriert werden :-)
- https://webpack.js.org/
- Seit 2012, MIT-Lizenz

Und Gatsby?

- Schnellst mögliche Website (blazing-fast)
  - Kurze Ladezeiten
  - Prefetching für schnelle Navigation in der Seite
- Erlaubt die Integration eigener Datenquellen
  - Dateien, existierende CMS, Datenbanken, APIs
  - Bereits viele existierende Module

How do you do?

Ein bisschen Praxis!

Let's do something

- Installation
- Einrichten Starter Projekt
- Anlegen einer Seite
- Datenabfrage mit GraphiQL
- Blog mittels Markdown-Dateien
- Styling mittels CSS Modulen

Einrichtung des Projekts

1. Installation:
   `npm install -g gatsby-cli`
2. Erstellen eines neuen Projekts:
   `gatsby new linuxwochen`
3. Wechsel in das Verzeichnis:
   `cd linuxwochen`
4. Start im Development Mode:
   `gatsby develop`
5. Git-Repository einrichten

Konfiguration für Debugging mit Chromium in VS Code:

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "type": "chrome",
            "request": "launch",
            "name": "Chrome gegen localhost
starten",
            "url": "http://localhost:8000",
            "webRoot": "${workspaceFolder}",
            "runtimeExecutable": "/usr/bin/chromium-
browser",
            "userDataDir": "/tmp/vscode-debug"
        }
    ]
}
```

Projektstruktur

- src
  - **pages** Seiten, dafür wird automatisch ein Link angelegt
  - **layouts** Seiten sind in einem Layout eingebaut
  - **components** Komponenten
- gatsby-config.js
- gatsby-node.js

Datenabfrage

- Einbau einer Abfrage in eine Seite
- Daten stehen in **props.data** zur Verfügung
- GraphiQL unter http://localhost:8000/___graphql

```
export const query =
graphql`
  query Page3Query {
    site {
      siteMetadata {
        title
      }
    }
  }
`
```

Plugins

Source Plugins

- Fügen zusätzliche Datenquellen ein

npm i -s gatsby-source-filesystem

gatsby-config.js

```
{
  resolve: `gatsby-source-filesystem`,
  options: {
    name: `src`,
    path: `${__dirname}/src/`,
  },
},
```

Transformer Plugins

- Verwandeln vorhandene Daten

npm i -s gatsby-transformer-remark

gatsby-config.js

```
`gatsby-transformer-remark`,
```

Erzeugen von Seiten (1/2)

1. Erweitern der Dokumentknoten um den Pfad
2. Anlegen einer Vorlage
3. Anlegen der Seiten

gatsby-node.js

```
const { createFilePath } = require('gatsby-source-
filesystem')

exports.onCreateNode = ({ node, getNode,
boundActionCreators }) => {
  const { createNodeField } = boundActionCreators
  if (node.internal.type === 'MarkdownRemark') {
    const slug = createFilePath({ node, getNode,
basePath: `pages` })
    createNodeField({
      node,
      name: 'slug',
      value: slug,
    })
  }
}
```

Erzeugen von Seiten (2/2)

src/templates/blog-post.js

```
import React from 'react'

export default ({ data }) => {
  const post = data.markdownRemark
  return (
    <article>
      <h1>{post.frontmatter.title}</h1>
      <div dangerouslySetInnerHTML={{ __html:
post.html }} />
    </article>
  )
}

export const query = graphql`
  query BlogPostQuery($slug:String!) {
    markdownRemark(fields: {slug: {eq: $slug}}) {
      frontmatter {
        title
      }
      html
    }
  }
`
```

gatsby-node.js

```
const path = require('path')

exports.createPages = async ({ graphql, boundActionCreators }) => {
  const { createPage } = boundActionCreators
  const { data } = await graphql(`
    {
      allMarkdownRemark {
        edges {
          node {
            fields {
              slug
            }
          }
        }
      }
    }
  `)
  data.allMarkdownRemark.edges.forEach(({ node }) => {
    createPage({
      path: node.fields.slug,
      component: path.resolve('./src/templates/blog-post.js'),
      context: {
        slug: node.fields.slug,
      },
    })
  })
}
```

Übersichtsseite

src/pages/index.js

```
import React from 'react'
import Link from 'gatsby-link'

const IndexPage = ({ data }) => {
  const posts = data.allMarkdownRemark
  return (
    <div>
      <h2>{posts.totalCount} Posts</h2>
      <ul>
        {posts.edges.map(({ node }) => (
          <li>
            <Link to={node.fields.slug}>
              <h3>
                {node.frontmatter.title}
                <small>{` ${node.timeToRead} minutes to read`}</small>
              </h3>
              <p>{node.excerpt}</p>
            </Link>
          </li>
        ))}
      </ul>
    </div>
  )
}

export default IndexPage
```

src/pages/index.js

```
export const query = graphql`
  query BlogPostsQuery {
    allMarkdownRemark {
      totalCount
      edges {
        node {
          fields {
            slug
          }
          frontmatter {
            title
          }
          excerpt
          timeToRead
        }
      }
    }
  }
`
```

Styling mit CSS Modulen

Änderung in *index.js*

```
import styles from './index.module.css'
…
    <div className={styles.index}>
…
```

Andere Varianten:

- CSS-in-JS
  - Glamor, Styled Components
- CSS-Präprozessoren
  - Sass, Styles, …

index.module.css

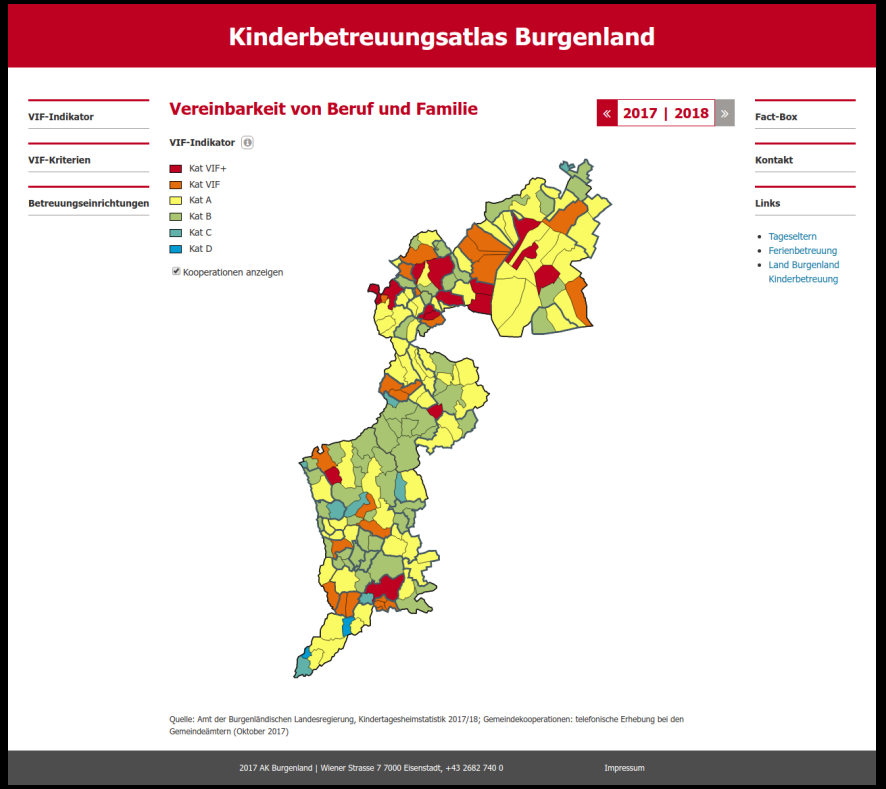
```
.index ul {
    list-style-type: none;
    padding: 0;
    margin: 0;
}

.index ul a {
    text-decoration: none;
    color: #111;
}

.index ul a h3 small {
    font-weight: normal;
}
```

Praxisbeispiel

- Daten aus Excel-Dateien nach JSON extrahiert
- Geo-Daten mittels D3.js aufbereitet
- SVG-Elemente in React implementiert
- https://www.kinderbetreuungsatlas.at/



Mehr Interesse?

franz@qnipp.com

qnipp