Wiki.js:

Pros:

* Nem at bruge for brugere der ikke er så teknisk faglige
* Hurtig opsætning
* Allerede integreret søge funktion(kan ændres til elastic search)
* Medie filer er nemme at tilføje
* Integration af Mermaid til at vise grafer
* Det er nemt at refererer til andre sider

Cons:

* Navigations systemet er lidt rodet, og det bliver kun værre når der kommer flere pages ind. Det vil være vigtigt at have god orden og standarder til hvordan nye pages bliver oprettet
* Det er ikke lige så nemt at referer til andre sider som den gamle wiki, og kan muligvis blive meget besværligt når der er mange sider.
* Hvis man har en reference til en anden side, og den side så bliver rykket, så stopper linket med at virke.

# Security:

* Lige nu står username og password til databasen bare som plaintext i et txt dokument. Skal finde en måde at kryptere det på eller bruge Integrated security.

[Node.js — Security Best Practices (nodejs.org)](https://nodejs.org/en/learn/getting-started/security-best-practices)

**Denial of service of the http server**

brug en reverse proxy. Det kan gøre DDOS angreb mindre effektive

**DNS Rebinding**

”do not run the inspector protocol in production” inspect protokollen bliver ikke brugt I programmet

**Exposure of sensitive information to an unauthorized actor**

Dette er et problem når man udgiver npm packages hvilket vi ikke gør

**HTTP request smuggling**

Et problem der kan udnyttes når front end og back end delen af et program håndterer HTTP requests forskelligt. Dette er et problem når man anvender insecureHTTPParser, hvilket programmet ikke gør.

**Information Exposure through timing attacks**

Når man indtaster et password som har den samme længde som det korrekte password tager det serveren en lille smule længere tid at sammenligne dem før den fortæller brugeren om det var korrekt eller forkert. Dette er et problem der kan ramme alle runtimes og ikke kun node.js

**Malicious third-party modules**

Third party packages kan være usikre. Så de kan være en sikkerhedsrisiko. For at køre et sikkerheds tjek på sine packages kan man kører

node --experimental-policy=policy.json app.js

når man loader dem.

**Supply chain attacks**

mange node.js packages har mange dependencies, som alle kunne blive compromised.

Man kan forhindre npm i at køre uønskede scripts med –ignore-scripts, og man kan tjekke efter kendte vulnerabilities med npm-audit.

Når man installere packages kan man bruge npm ci i stedet for npm install. Det forhindrer inkonsistensens mellem package.json og lockfile

**Monkey patching**

Forstår ikke

**Prototype pollution attacks**

Forstår ikke

Security fra wiki.js

A screenshot of a computer

Description automatically generated

# Graph QL:

* Skulle være hurtigere og mere effektivt end REST
* Mere fleksibelt når man henter data
* Kan bruges med mange front end frameworks

# Uge 31

* Custom tags eller findes de allerede med markdown ([Wiki Demo (orifarm.net)](http://wiki.orifarm.net/wiki/650455/wiki-demo))
* Syntax highlighting for code
* Kig på hvilke moduler der er relvate f.eks mermaid
* Kan man tilføje flere moduler?
* Forstå search engine og kig på hvad de andre kan
* Find og forstå koden bagved import everything
* Kig mere på graphql
* Customization af wikien
* Få et overblik over cache
* Når man injecter sider hvordan fungerer det så i forhold til forskellige sprog
* Wiki customization [GitHub - madodig/wikijs-customization: Wiki.js Customization Injection Code](https://github.com/madodig/wikijs-customization)
* WikiJS (hacks) [WikiJS hacks | MakersMashup Wiki](https://wiki.makersmashup.com/en/other-stuff/wikijs)
* Kan man kalde funktioner som import all på andre måder
* Find en nemmere måde at link google maps på wikien, custom css code og html injection

Databases>master>programmability>functions>scalar-valued functions

Eksempler:

SELECT \* FROM file\_system.get\_drive\_info()

SELECT file\_system.read\_text(N'C:\wiki\backup\home.md', DEFAULT)

SELECT [master].[file\_system].[write\_text](N'C:\wiki\backup\test2.md', N'Dette er en demo fff', N'UTF-8')

SELECT regex.find(N'dette er et tal 1238555', N'\d+',N'0',3)

# Page Linking noter

Når man linker til en anden side gør man det med følgene syntaks

(titel)[link]

1. de indbyggede metode

når man editer et page så kan man bruge ”insert link” knappen til referer til en anden siden

A screenshot of a computer

Description automatically generated

Dette er en nem og intuitiv måde at gøre det på.

Der kan opstå problemer jo større wikien er. Det kan blive svært at finde rundt i alle mapperne jo flere af dem der er.

Når man skriver path’en ind til et link så kommer den med forslag

A screen shot of a computer

Description automatically generated

1. manuel linking

når man gerne vil referer en side kan man gøre det bare ved at skrive titlen på den f.eks:

[Embed](./Embed). Hvor teksten er ”Embed” og den refferer til siden ”Embed” som ligger i samme mappe som den nuværende fil der er ved at blive redigeret.

Denne løsning virker kun hvis alle sider ligger i den samme mappe. Dette kan skabe problemer, med navigation på wikien og det kan gøre det sværere at sætte page permissions til forskellige grupper af folk. Det nemmeste er at sætte permissions på forskellige mapper. Hvis der ikke er nogle mapper kan man sætte permissions på specifikke pages eller ud fra tags.

# Custom tags noter:

**Underline**: kan slås til i settings, men er ummidelbart reserveret til noget andet

**Table of contents**: den bliver autogenereret ud fra headers.

**Youtube link**: der er implementeret en løsning med code injection, men custom code kan få sider til at load langsommere

**Maps**: man kan linke med html kode kopieret direkte fra google maps, men det er en besværlig løsning

**Flag**: man kan bruge emojis.

**Billede af medarbejdere**: Man kan bruge normal insert billede, men kan ikke komme på en bedre metode før ldap er sat op

**Display box**: >Your text here {.is-info} eller anvend knap fra toolbar

**Highlighting**: ==Your text here==

**Linking**: læs tidligere afsnit

**Table**: anvend knap fra toolbar i visual editor eller

| **Header 1** | **Header 2** | **Header 3** |

|----------|----------|----------|

| Foo | Bar | Xyz |

| Abc | Def | 123 |

**Syntax highlighting**:[[1]](#footnote-1)

```<sprog>

// some code here

```

Et eksempel:

```SQL

SELECT \* FROM file\_system.get\_drive\_info()

SELECT file\_system.read\_text(N'C:\wiki\backup\home.md', DEFAULT)

SELECT [master].[file\_system].[write\_text](N'C:\wiki\backup\test2.md', N'Dette er en demo fff', N'UTF-8')

SELECT regex.find(N'dette er et tal 1238555', N'\d+',N'0',3

```

Hvis man ikke vil have linje nummer med på sin kode kan man tilføje følgende <div> klasse inden sin kode

<div class="next-codeblock-no-line-numbers"></div>

# Elastic Search vs Regular

Regular search kan kun søge i titler på pages men elastic search kan søge efter ord i pages.

# Elastic Search vs Azure Search

## Azure

* Støtter regular expressions
* Kan søge efter billeder med hjælp af ai (ved ikke hvor godt det virker)
* Man kan søge inde i filer som word, excel, PDF og HTML
* Geo-search. Den viser automatisk resultater som er mere relevant i forhold til din geografiske lokation. Kunne være relevant hvis folk i andre lande skal bruge wikien.
* Bruger venligheds features. Auto-complete, search suggestions, synonymer,
* Search analytics

## ElasticSearch

* Er fokuseret på “document” søgning
* Textual search: kan søge efter tekst inde i dokumenter/pages
* Geo-search. Den viser automatisk resultater som er mere relevant i forhold til din
* Bruger venligheds features. Auto-complete, search suggestions

Find en made at kald på rebuild page tree. Graphql eller en url

1. [highlight.js/SUPPORTED\_LANGUAGES.md at main · highlightjs/highlight.js · GitHub](https://github.com/highlightjs/highlight.js/blob/main/SUPPORTED_LANGUAGES.md) – alle støttede sprog til syntax highlighting [↑](#footnote-ref-1)