Zpravodaj Československého sdružení uživatelů TeXu

Michal Hoftich

Publikace LaTeXových dokumentů na webu pomocí TeX4ht a GitHub Actions

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 35 (2025), No. 1-4, 11–25

Persistent URL: http://dml.cz/dmlcz/153196

Publikace LaTeXových dokumentů na webu pomocí TeX4ht a GitHub Actions

Michal Hoftich

Článek představí sadu šablon pro nástroj TeX4ht, který slouží k převodu LaTeXových dokumentů do HTML. Tyto šablony výrazně usnadňují publikaci různých typů dokumentů na webu a přinášejí možnosti zpracování a automatizace.

První šablona je určena pro převod knižních dokumentů do webové podoby. Umožňuje rozdělení textu do jednotlivých kapitol s automaticky generovanou navigací a podporou responzivního designu, takže je výsledek dobře čitelný i na mobilních zařízeních.

Druhá šablona slouží k tvorbě staticky generovaných blogů. Každý příspěvek je psán jako samostatný LaTeXový dokument, který je pomocí TeX4ht převeden do HTML. Následně jsou tyto články zpracovány statickým generátorem webů, jako je například Jekyll, který se postará o sestavení celého blogu, vytvoření rozcestníků, archivů a další navigace.

Třetí šablona je zaměřena na převod prezentací vytvořených v prostředí Beamer do formy tzv. handoutů – přehledových materiálů pro posluchače. Výsledkem je čitelný a dobře strukturovaný webový dokument vhodný pro sdílení po přednášce.

Všechny šablony jsou navrženy tak, aby fungovaly v rámci GitHub Actions. To znamená, že dokumenty mohou být automaticky zkompilovány a publikovány online pokaždé, když dojde ke změně v repozitáři.

Klíčová slova: TeX4ht, LaTeX, HTML, Jekyll, Beamer, GitHub Actions

1. Úvod do TeX4ht

TeX4ht je nástroj pro konverzi z LaTeXu do HTML a dalších formátů, jako jsou ODT, EPUB nebo JATS XML. Dokáže zachovat strukturu i formátování původního dokumentu a nabízí různé metody pro matematický výstup, včetně obrázků, MathML a MathJaxu. Umožňuje také generovat obrázky přímo z výstupu LaTeXu a podporuje běžně používané LaTeXové balíčky.

TeX4ht samotný je především balíček, který redefinuje příkazy jiných balíčků tak, aby vkládaly tagy výstupních formátů. Části DVI výstupu lze konvertovat na obrázky ve formátu PNG nebo SVG. Toho se využívá pro podporu obrázků tvořených pomocí balíčků Ti*k*Z nebo PSTricks.

Více informací o TeX4ht a jeho základním použití naleznete v mém starším článku ve *Zpravodaji* [1]. V tomto článku se zaměřím na pokročilejší možnosti vyu-

DOI: 10.5300/2025-1-4/11

žití TEX4ht a na představení šablon pro publikaci dokumentů na webu s využitím GitHub Actions.

2. Automatické generování HTML výstupu pomocí GitHub Actions

2.1. Základní struktura konfigurace GitHub Actions

GitHub, ale i další repozitáře jako GitLab nebo Bitbucket, umožňuje spouštět definované scénáře (např. build, testy nebo nasazení) v reakci na události v repozitáři (push, pull request apod.).

Tato funkce se nazývá GitHub Actions [2]. V tomto článku si ukážeme, jak ji využít pro automatizovanou kompilaci dokumentů pomocí TEX4ht a jejich publikování na webu.

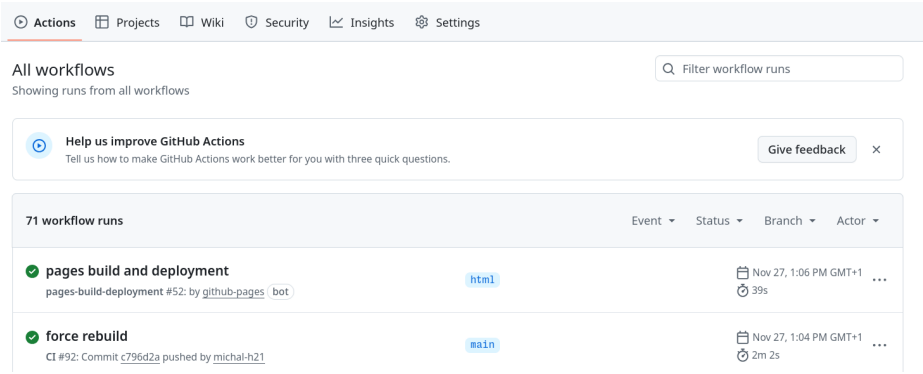
Šablony pro TEX4ht, které dále představím, jsou navrženy tak, aby fungovaly v rámci GitHub Actions. To znamená, že dokumenty mohou být automaticky zkompilovány a publikovány online pokaždé, když dojde ke změně v repozitáři. Tento přístup zajišťuje, že webová verze dokumentu je vždy aktuální.

Sada pravidel, která definují, jaké akce se mají provést v návaznosti na nějakou událost v repozitáři, se v terminologii GitHub Actions nazývá *workflow*. Workflow je definován v souboru ve formátu YAML, který je po stažení šablon dostupný v podadresáři `.github/workflows/main.yml`. YAML je jednoduchý textový formát používaný pro konfigurace. GitHub Actions v něm definují jednotlivé kroky automatizace. Například v šabloně pro „handouty“ prezentací se používá následující kód:

```
- name: Spuštění make4ht
  uses: xu-cheng/texlive-action/full@v1
  with:
    run: |
      make4ht -lj index -a debug -d out handout.tex

- name: Publikování webových stránek
  uses: peaceiris/actions-gh-pages@v3
  with:
    github_token: ${{ secrets.GITHUB_TOKEN }}
    publish_dir: ./out
```

Jednotlivé kroky úlohy jsou definovány v sekci `steps`, jejíž obsah zobrazuje předešlý úryvek kódu. Každý krok provede určitou akci, například kompilaci LATEXu. K dispozici má soubory z repozitáře a také soubory vytvořené v předešlých krocích. V této ukázce se nachází dva kroky, které používají dvě GitHub Actions – `xu-cheng/texlive-action` [3] a `peaceiris/actions-gh-pages` [4]. První z nich umožňuje používat libovolný příkaz dostupný v instalaci TEX Live jako `make4ht`



Obrázek 1: Rozhraní GitHub Actions zobrazující stav workflow

nebo `lualatex`. Druhá publikuje obsah zadaného adresáře na webu pomocí GitHub Pages.

Pro spuštění akce je třeba pouze nahrát změny do repozitáře pomocí následujících příkazů:

```
$ git add ⟨upravené soubory⟩
$ git commit -m "Popis změn"
$ git push origin main
```

Příkaz `git add` přidá upravené soubory do indexu pro commit, `git commit` vytvoří nový commit s popisem změn a `git push` nahraje změny do větve `main` na vzdáleném repozitáři. Při každé změně v repozitáři se poté spustí workflow, který zkompiluje soubor `handout.tex` do HTML pomocí příkazu `make4ht`:

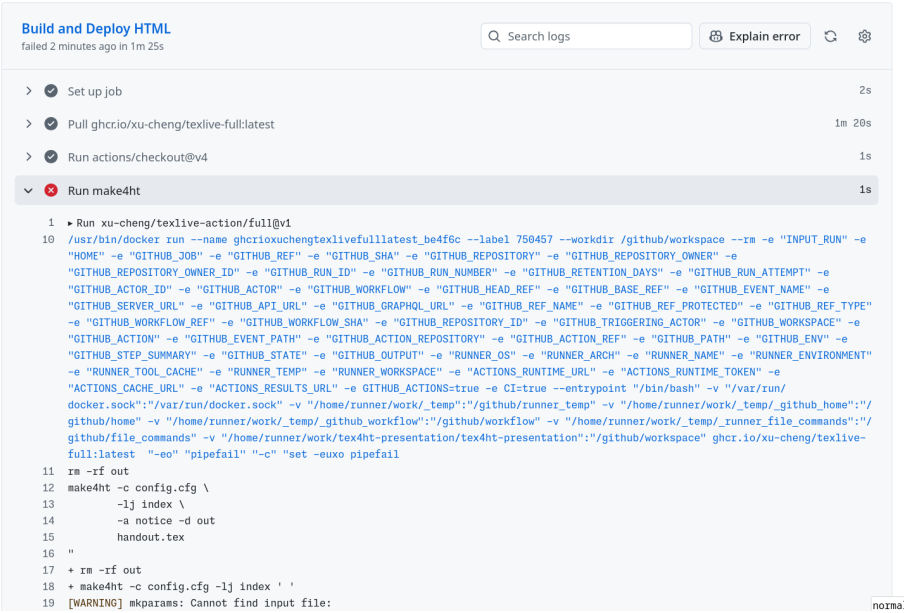
```
$ make4ht -lj index -a debug -d out handout.tex
```

Tento příkaz vytvoří HTML soubory v adresáři `out` pomocí volby `-d out`. Volba `-lj index` je zkratkou pro volby `--lua`, která vyvolá kompilaci LuaTEXem, a `--jobname index`, která nastaví název výstupního HTML souboru na `index.html`. Akce `peaceiris/actions-gh-pages` poté zveřejní všechny soubory v adresáři `out`, což je specifikované nastavením vlastnosti `publish_dir` ve workflow souboru.

HTML verze prezentace bude dostupná na následující adrese:

`https://`⟨*jméno uživatele*⟩`.github.io/`⟨*název repozitáře*⟩`/`

V URL není třeba specifikovat název souboru – GitHub Pages automaticky nabízí soubor `index.html`. Tato funkce usnadňuje sdílení prezentace a pomáhá předejít nefunkčním odkazům kvůli neshodě názvů souborů.



Obrázek 2: Příklad chyby v GitHub Actions workflow

2.2. Sledování stavu workflow a řešení chyb

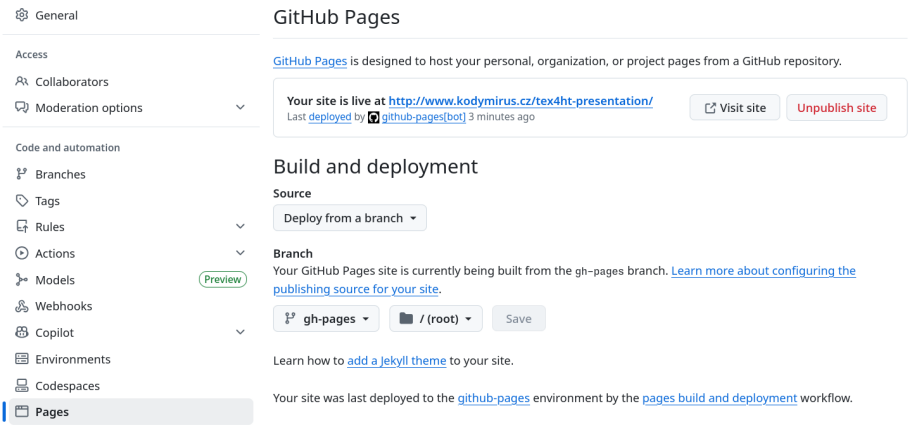
Po každém nahrání změn můžete zkontrolovat záložku `Actions` ve svém GitHub repozitáři (vizte Obrázek 1). Zobrazuje stav workflow včetně informace o úspěšném dokončení nebo případných chybách.

Můžete také zkontrolovat logy běhu workflow, abyste viděli, co se pokazilo. Pokud narazíte na chybu, bude zobrazena v logu a tyto informace můžete použít k řešení problému.

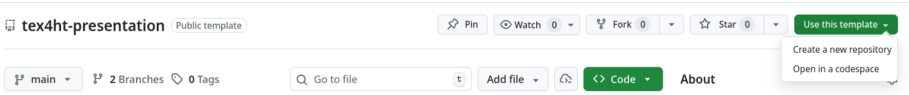
Například v logu na Obrázku 2 vidíme, že u kroku `Run make4ht` se zobrazila červená výstražná ikonka. Po kliknutí na název kroku se zobrazí podrobnosti o chybě. V tomto případě byl nesprávný název hlavního TEXového souboru, který se kompiloval pomocí `make4ht`, což nám oznamuje hláška „`make4ht: missing required parameter: filename`“. V YAML souboru GitHub Actions tedy musíme opravit název souboru na správný.

2.3. Publikování HTML verze

Po úspěšném dokončení workflow můžete nastavit GitHub Pages pro zobrazování obsahu větve `gh-pages`. Výstupní soubory vytvořené pomocí `make4ht` budou dostupné na adrese `https://`⟨*jméno uživatele*⟩`.github.io/`⟨*název repozitáře*⟩`/`.



Obrázek 3: Nastavení GitHub Pages pro repozitář

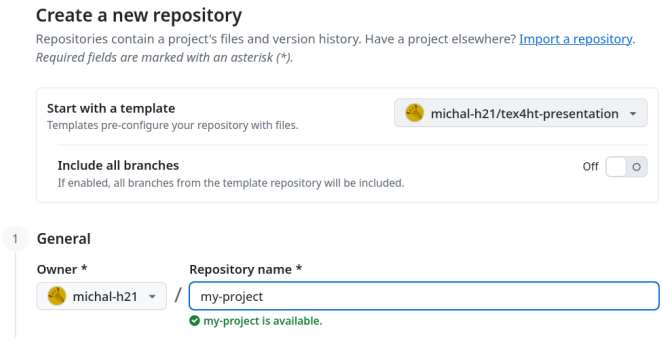


Obrázek 4: Tlačítko pro použití šablony na GitHubu

Aby se HTML verze dokumentu zobrazila správně, je třeba v nastavení GitHub repozitáře nastavit GitHub Pages tak, aby používala větev `gh-pages` jako zdroj pro publikování (vizte Obrázek 3). Tuto větev používá akce `peaceiris/actions-gh-pages` pro publikování výstupních souborů. Díky tomu, že se používá samostatná větev, zůstává hlavní větev `main` čistá a obsahuje pouze zdrojové TEXové soubory.

2.4. Využití šablon na GitHubu

Pro využití šablon na GitHubu klikněte na tlačítko „Use this template“ na stránce GitHub repozitáře s podporou šablon (vizte Obrázek 4). Po vyplnění dialogu pro vytvoření nového repozitáře z šablony se ve vašem účtu vytvoří nový repozitář se stejnou strukturou a se stejnými soubory, jako měl původní repozitář, ovšem bez jeho Git historie. Poté nově vytvořený repozitář můžete naklonovat na svůj počítač a začít upravovat obsah souborů.



Obrázek 5: Dialog vytvoření nového repozitáře ze šablony

3. Šablona pro webové verze knih

Tato šablona [5] je navržena tak, aby umožňovala export každé kapitoly do samostatné HTML stránky. Je ideální pro publikaci knihy, dokumentace nebo skript, kde každá kapitola tvoří vlastní webovou stránku propojenou navigací.

Pokud chceme vytvořit samostatné HTML soubory pro jednotlivé kapitoly našeho dokumentu, můžeme použít následující příkaz:

```
$ make4ht document.tex "2"
```

Číselné volby umožňují nastavit, od které úrovně členění textu se mají vytvářet samostatné soubory. Například volba s hodnotou 1 způsobí, že budou vytvořeny samostatné soubory pro nejvyšší úroveň struktury dokumentu. Hodnota 2 zajistí rozdělení i pro další úroveň, obvykle kapitoly, případně sekce, pokud typ dokumentu kapitoly neobsahuje. Při volbě 3 se soubory vytvářejí i pro sekce, a to i u typů dokumentů, které běžně kapitoly obsahují. Takto lze pokračovat až do úrovně 7, kdy jsou samostatné soubory generovány i pro velmi jemné členění, na úrovní příkazů `\paragraph`. Obecně doporučuji použít volbu `"2"`.

3.1. Pojmenování souborů kapitol

Ve výchozí podobě jednotlivé soubory kapitol obsahují jen základní navigaci s odkazy na předchozí a následující kapitolu a na hlavní soubor. Názvy souborů jsou tvořeny názvem hlavního souboru, následovaným zkratkou úrovně nadpisu a pořadím dané úrovně nadpisu v dokumentu. Například pro dokument s názvem `document.tex` budou názvy souborů pro jednotlivé kapitoly `documentch1.html`, `documentch2.html` a soubor s obsahem, titulní stránkou a textem, který se nenachází uvnitř žádné kapitoly, bude pojmenovaný `document.html`. Pro nečíslované kapitoly, například rejstřík nebo seznam literatury, se vytváří soubor s koncovkou `li`, například `documentli1.html`.

Název kapitoly	výchozí název	sec-filename	sec-slugname
S háčky	`documentch1.html`	`Sháčky.html`	`s_hacky.html`

Tabulka 1: Názvy souborů kapitol podle různých způsobů pojmenování

Problém nastane, pokud změníme pořadí kapitol nebo přidáme kapitolu novou. Odkazy z externích webů na sekce jednotlivých kapitol mohou přestat fungovat, nebo budou odkazovat jinam, protože se jejich obsah přesune do jiného souboru. Abychom tomu předešli, můžeme využít jiné způsoby pojmenování souborů. TEX4ht nabízí dvě možnosti, jak pojmenovat soubory na základě názvů kapitol. První z nich je `sec-filename`, druhá `sec-slugname`. Jejich použití ukazuje Tabulka 1. Z hlediska použitelnosti na webu je lepší volba `sec-slugname`. Pro svou funkcionalitu ovšem vyžaduje přepínač `-l`, protože název se vytváří pomocí jazyka Lua. Volby pro `make4ht` budou tedy následující:

```
$ make4ht -l document.tex "2,sec-slugname"
```

3.2. Boční menu s obsahem

Problémem aktuálně vytvořených HTML stránek je, že neobsahují navigační menu s odkazy na všechny kapitoly. Navigace obsahuje pouze odkazy na předchozí a následující kapitolu a kořenový soubor. Pro snadnou navigaci mezi kapitolami můžeme využít volbu `fulltoc`. Ta vytvoří na každé stránce boční menu obsahující plný obsah knihy (vizte Obrázek 6a). Díky tomu lze snadno přecházet mezi jednotlivými kapitolami.

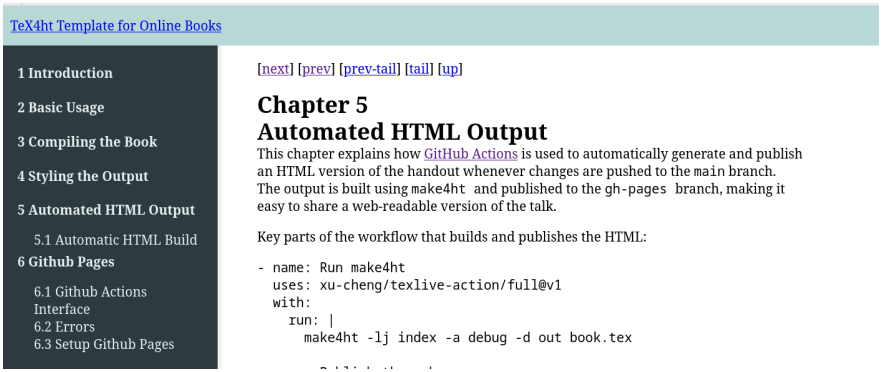
Ovšem toto menu může být poměrně rozsáhlé, pokud je v knize hodně podkapitol. Můžeme chtít zobrazit pouze sekce aktuální kapitoly a sekce ostatních kapitol skrýt (vizte Obrázek 6b). Toho je možné docílit pomocí JavaScriptu nebo DOM filtru `collapsetoc` v `make4ht`, což je možnost, kterou šablona využívá.

Skrytí menu je užitečné především na mobilních zařízeních s menším displejem, kde by při zobrazeném menu nezůstalo dostatek místa pro zobrazení samotného textu dokumentu. Pro zobrazení menu využíváme JavaScript, kterému se jinak snažíme vyhýbat.

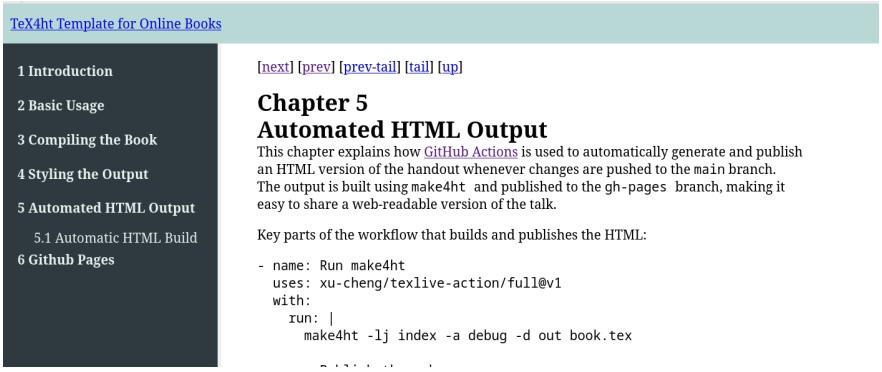
3.3. Konfigurační soubor

Většina barev v šabloně je nastavená pomocí CSS proměnných, které můžeme změnit v konfiguračním souboru `config.cfg`. K dispozici jsou následující vlastnosti, které můžeme upravit:

maintoc – menu s obsahem dokumentu,
hamburger – tlačítko pro skryté menu,



(a) Ukázka bočního menu s obsahem dokumentu



(b) Ukázka bočního menu s rozbalenými sekcemi pouze aktuální kapitoly

Obrázek 6: Porovnání bočního menu s plným obsahem dokumentu a menu s rozbalenými sekcemi pouze aktuální kapitoly

header – hlavička dokumentu,
footer – patička dokumentu.

Pro každou z těchto vlastností můžete nastavit barvu popředí a pozadí přidáním `-color`, respektive `-background` k jejich názvu.

Například pro změnu barev bočního menu můžeme použít následující kód:

```
\Css{
  body{
    --maintoc-background: \#f0f0f0;
    --maintoc-color: \#00AFA0;
  }
}
```

Příkaz `\Css` slouží pro vložení kódu do CSS souboru, který TEX4ht vytváří. Můžete jej vložit do konfiguračního souboru `config.cfg` kamkoliv mezi příkazy `\Preamble` a `\EndPreamble`. Při použití hexadecimálního zápisu je třeba escapovat znak `#` pomocí zpětného lomítka, jinak dojde ke kompilační chybě.

4. Šablona pro blogování s LATEXem

Tato šablona [6] umožňuje vytvářet blog pomocí statických generátorů webů. Využívá přitom TEX4ht pro generování HTML verze jednotlivých článků ve formátu, který je kompatibilní s většinou statických generátorů webů, jež podporují HTML soubory jako vstupní formát.

Statické generátory webů představují způsob, jak vytvářet a spravovat webové stránky bez nutnosti provozovat databázi nebo dynamický backend. Obecně fungují tak, že z jednoduchých textových souborů (například ve formátu Markdown nebo HTML) vytvoří kompletní webové stránky pomocí šablon a dalších pravidel. Vytvářejí také archivy článků, RSS kanály a další doprovodné funkce, které jsou běžné na blozích.

Výhodou je, že takový web můžeme umístit na téměř libovolný webový server; nemusíme řešit, zda podporuje například PHP nebo jiný systém pro dynamické zobrazování stránek. Například GitHub nabízí GitHub Pages, kam můžeme umístit stránky zdarma. V rámci GitHub Actions také můžeme spouštět generátory automaticky, nebo můžeme využít vestavěnou podporu pro generátor Jekyll.

V naší šabloně je každý článek vytvořen jako samostatný TEXový dokument umístěný v odděleném adresáři. Využíváme skript, který prochází všechny adresáře s články, dokumenty, které se od minulé kompilace změnily, kompiluje je pomocí TEX4ht a ukládá vygenerované HTML soubory do adresáře, který slouží jako vstup pro statický generátor webu. Tyto soubory jsou automaticky uloženy do zvláštní větve GitHub repozitáře, takže výstupní HTML soubory nemusíme spravovat ručně. Ukládáme je proto, abychom nemuseli při každém sestavení blogu znovu kompilovat všechny články, ale pouze ty, které se od minulé kompilace změnily.

4.1. Formát vstupních souborů

Statické generátory webů obvykle vyžadují, aby vstupní soubory obsahovaly blok s metadaty v následujícím formátu:

```
---
title: Titulek dokumentu
---
# Úvod
Text dokumentu
```

Na začátku souboru je umístěn blok s metadaty dokumentu ve formátu YAML. Z obou stran je obklopený řádky obsahujícími pouze tři spojovníky. Za blokem metadat pak následuje text dokumentu.

V tomto příkladě používáme Markdown, ovšem stejný princip můžeme použít i pro HTML soubory produkované pomocí TEX4ht. Místo Markdownu totiž můžeme použít i HTML, které získáme z obsahu elementu `<body>`.

Příkaz `make4ht` obsahuje rozšíření `staticsite`, které umožňuje generovat HTML soubory s YAML metadaty z TEXových dokumentů. Automaticky například přidává titulek dokumentu získaný z příkazu `\title` a další metadata. Mějme například následující jednoduchý TEXový dokument:

```
\documentclass{article}
\begin{document}
\title{Hello world test}
\author{Michal}
\maketitle
This is my test post.
\end{document}
```

Tento dokument můžeme zkompilovat do HTML s YAML preamble pomocí následujícího příkazu:

```
$ make4ht -f html5+staticsite filename.tex
```

Rozšíření `staticsite` zapneme tím, že jeho název připojíme znaménkem plus za formát souboru ve volbě `-f`. Soubor bude automaticky pojmenován ve formátu `YYYY-MM-DD-`⟨*název originálu*⟩. Vygenerovaný soubor bude vypadat následovně:

```
---
meta:
- charset: 'utf-8'
- name: 'viewport'
  content: 'width=device-width,initial-scale=1'
time: 1626619562
updated: 1627244699
styles:
- '2021-07-18-hello-world.css'
title: 'Hello world test'
```

```
---
<!--  l. 7  --><p class='indent'>   This is my test post.
</p>
```

Soubor obsahuje různá metadata získaná z HTML souboru. Jednak se jedná o obsah elementů `<meta>`, dále časy vytvoření souboru a poslední aktualizace v číselné formě, seznam použitých stylů, titulek atd. Tato metadata můžeme používat v šablonách použitého generátoru webů.

4.2. Struktura šablony pro blog

Šablona má následující strukturu adresářů, která předpokládá použití statického generátoru Jekyll [7]:

```
blog/
.. texposts/
.... first_post/
...... first_post.tex
.... second_post/
...... second_post.tex
.. docs/
.... _posts/
.... css
```

Každý příspěvek na blogu má vlastní podadresář v adresáři `texposts`. Vygenerované soubory se poté zkopírují do různých podadresářů v adresáři `docs`, který Jekyll použije jako zdrojový adresář pro sestavení blogu. Například HTML soubory pro blog se umístí do podadresáře `_posts` a kaskádové styly do podadresáře `css`.

Pro start nového blogu je třeba odstranit všechny podadresáře v adresáři `texposts` kromě adresáře `cookiecutter-post`, který slouží jako šablona pro nové příspěvky.

```
$ find texposts -mindepth 1 -maxdepth 1 -type d \
  ! -name cookiecutter-post \
  -exec rm -rf {} +
$ git add -u texposts
$ git commit -m "Clean start for a new blog"
$ git push
```

Nové příspěvky pak můžeme přidávat ručně vytvořením podadresáře v adresáři `texposts`, nebo využít šablonu `cookiecutter-post` pomocí nástroje Cookiecutter [8]:

```
$ cd texposts
$ cookiecutter cookiecutter-post
```

Nástroj se zeptá na název příspěvku, autora a další informace a poté vytvoří nový adresář se základním TEXovým souborem. Pro změnu výchozích hodnot můžeme

upravit soubor `cookiecutter.json` v adresáři `cookiecutter-post`, TEXovou šablonu pak můžeme upravit v souboru `{{cookiecutter.project_slug}}` ve stejném adresáři.

Po dokončení úprav nového příspěvku jej můžeme přidat do repozitáře, ovšem je třeba také vytvořit soubor, který obsahuje timestamp publikace příspěvku. Tento soubor musí mít název ⟨*jméno dokumentu*⟩`.published` a také je třeba přidat jej do repozitáře. Na základě timestampu se vytváří neměnná URL příspěvku, neboť Jekyll používá datum v názvu souboru pro generování URL. Soubor můžeme vytvořit například pomocí následujícího příkazu:

`$ date +%s >` ⟨*jméno dokumentu*⟩`.published`

Další možností je zkompilovat dokument pomocí `make4ht` s využitím módu `publish`:

`$ make4ht -m publish` ⟨*jméno dokumentu*⟩`.tex`

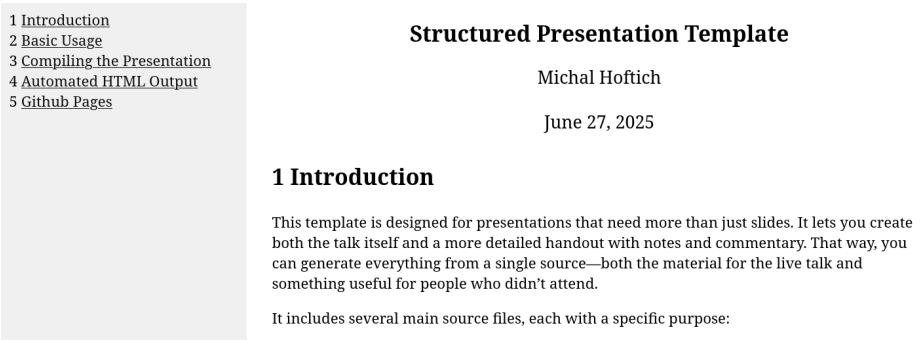
Tento příkaz vytvoří soubor s timestampem publikace automaticky a navíc zkopíruje vygenerované HTML nebo CSS soubory do vstupního adresáře pro Jekyll.

Po přidání nového příspěvku a souboru `.published` do repozitáře se automaticky spustí GitHub Actions workflow, který aktualizuje blog.

5. Šablona pro prezentace

Tato šablona [9] je určena pro prezentace, které potřebují víc než jen samotné snímky. Umožňuje vytvořit nejen prezentaci, ale také textovou verzi prezentace ve formátu HTML (vizte Obrázek 7). Šablona automaticky vytváří HTML verzi článku, která je poté zveřejněna na GitHub Pages. Veškeré materiály se generují z jednoho zdrojového souboru, čímž se zjednodušuje správa celého projektu a zajišťuje se konzistence mezi jednotlivými výstupy.

Součástí projektu je několik souborů, které jsou používané pro generování různých výstupů prezentace. Soubor `slides.tex` slouží ke generování hlavní prezentace ve formátu Beamer a obsahuje vše, co se zobrazuje během přednášky. Pro vytvoření textové, čtenářsky přívětivé verze prezentace je určen `handout.tex`, který je formátován jako klasický článek. Kromě obsahu viditelného na snímcích zahrnuje i doplňující poznámky a komentáře a je koncipován tak, aby byl samostatně srozumitelný i pro ty, kteří prezentaci neviděli. Samotný obsah prezentace je uložen v souboru `presentation.tex`. Text uvnitř prostředí `\begin{frame}...\end{frame}` se vkládá jak do prezentace, tak do článku, zatímco materiál mimo toto prostředí se nezobrazuje na snímcích, ale zůstává součástí textové verze. Díky tomu může článek obsahovat podrobnější vysvětlení nebo komentáře bez narušení struktury prezentace. Soubory `preamble.tex` a `config.cfg` doplňují konfiguraci: první z nich obsahuje balíčky a definice příkazů používané v prezentaci a sdílené v obou výstupech, zatímco druhý slouží jako



Obrázek 7: Ukázka článku z prezentace v HTML formátu

konfigurační soubor pro TEX4ht a umožňuje upravit například CSS stylování webové verze nebo redefinice LATEXových příkazů.

Obecně by mělo stačit upravovat obsah souboru `presentation.tex`, který obsahuje jak slidy pro Beamer, tak i poznámky pro článek. V souboru `preamble.tex` můžeme přidat další balíčky nebo vlastní příkazy a v `config.cfg` můžeme upravit styly webové verze článku.

Obsah slidu s textovým popisem pro článek může vypadat například takto:

```
\begin{frame}[fragile]{Nadpis slidu}
\begin{itemize}
\item bod 1
\item bod 2
\end{itemize}
\end{frame}

Zde je textový popis mimo prostředí \texttt{frame}.
Zobrazí se pouze ve článku.
```

Obsah uvnitř bloku `\begin{frame}...\end{frame}` se zahrne do prezentace i článku, ale následující odstavec – mimo toto prostředí – se zobrazí pouze v článku.

Pokud chceme obsah mimo prostředí `frame` zahrnout jak do prezentace, tak do článku, můžeme použít příkaz `\mode`:

```
\mode<beamer|article>{
\title{Strukturovaná šablona prezentace}
\author{Michal Hoftich}
\maketitle}
```

Příkaz `\mode<beamer|article>{...}` umožňuje zahrnout obsah, který se má zobrazit jak v prezentaci, tak v handoutu. To se může hodit například pro titulní stranu nebo obsah prezentace.

Odkazy

1. HOFTICH, Michal. Publikování z LaTeXu na web pomocí TeX4ht. *Zpravodaj CSTUG*. 2018, **28**(1), 11–21. Dostupné z DOI: 10.5300/2018-1-4/11.
2. GITHUB. *Quickstart for GitHub Actions* [online]. 2025. [cit. 2025-11-21]. Dostupné z: https://docs.github.com/en/actions/get-started/quickstart.
3. XU-CHENG. *texlive-action: GitHub Action to run arbitrary commands in a TeXLive environment* [online]. 2025. [cit. 2025-11-21]. Dostupné z: https://github.com/xu-cheng/texlive-action.
4. UEDA, Shohei; PEACEIRIS. *actions-gh-pages: GitHub Actions for GitHub Pages (deploy static files)* [online]. 2025. [cit. 2025-11-21]. Dostupné z: https://github.com/peaceiris/actions-gh-pages.
5. HOFTICH, Michal. *tex4ht-booksite: Template for web versions of LaTeX books* [online]. 2025. [cit. 2025-11-21]. Dostupné z: https://github.com/michal-h21/tex4ht-booksite.
6. HOFTICH, Michal. *Testblog: Blogging template for TeX4ht* [online]. 2025. [cit. 2025-11-21]. Dostupné z: https://github.com/michal-h21/testblog.
7. *Jekyll: Simple, blog-aware, static sites* [online]. 2025. [cit. 2025-11-21]. Dostupné z: https://jekyllrb.com/.
8. *Cookiecutter: A command-line utility that creates projects from project templates* [online]. 2025. [cit. 2025-11-21]. Dostupné z: https://github.com/cookiecutter/cookiecutter.
9. HOFTICH, Michal. *tex4ht-presentation: Template for web versions of LaTeX Beamer presentations* [online]. 2025. [cit. 2025-11-21]. Dostupné z: https://github.com/michal-h21/tex4ht-presentation.

Summary: Publishing LaTeX Documents on the Web Using TeX4ht and GitHub Actions

The article presents a set of templates for the TeX4ht tool, which serves to convert LaTeX documents into HTML. These templates greatly simplify the publication of various types of documents on the web and bring modern capabilities for processing and automation.

The first template is intended for converting book-style documents into a web format. It allows the text to be divided into individual chapters with automatically generated navigation and support for responsive design, making the result easily readable even on mobile devices.

The second template is used for creating statically generated blogs. Each post is written as a separate LaTeX document, which is converted to HTML using

TEX4ht. Subsequently, these articles are processed by a static site generator such as Jekyll, which takes care of assembling the entire blog, creating indexes, archives, and other navigation elements.

The third template focuses on converting presentations created in the Beamer environment into so-called handouts – overview materials for the audience. The result is a readable and well-structured web document suitable for sharing after the lecture.

All templates are designed to work within GitHub Actions. This means that the documents can be automatically compiled and published online whenever a change is made in the repository. This approach ensures that the web version of the document is always up to date.

Keywords: TEX4ht, LATEX, HTML, Jekyll, Beamer, GitHub Actions

Michal Hoftich, michal.h21@gmail.com