Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

HTML5 – struktura a sémantika

Josef Vegner

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Josef Vegner

Informatika

Název práce

HTML5 – struktura a sémantika

Název anglicky

HTML5 – structure and semantics

Cíle práce

Cílem této bakalářské práce je poukázat na možnosti použití a výhody, které přináší nová sémantika v HTML5. Práce bude zaměřena na nové elementy, jejich podrobné vysvětlení a využití v praxi. Důraz bude kladen především na vzájemné propojení HTML5 elementů s technologií WAI-ARIA.

Metodika

Řešení problematiky bakalářské práce bude založeno na studiu a analýze odborných informačních zdrojů. Teoretická část bude věnována především vysvětlení jednotlivých pojmů, správné HTML5 syntaxi a struktuře. Praktická část bude důkladně analyzovat sémantiku vytvořené webové stránky. Tato webová stránka bude vytvořena celkem třikrát. První varianta bude obsahovat výhradně elementy HTML5, druhá varianta bude tvořena pomocí technologie WAI-ARIA a třetí varianta bude kombinovat výhody obou technologií. Součástí praktické části bude také podrobná analýza podpory použitých elementů a vlastností webovými prohlížeči. Výsledky práce budou přehledně vyhodnoceny a interpretovány.

Doporučený rozsah práce

30 – 40 stran

Klíčová slova

html5, wai-aria, sémantika, přístupnost, web, standard, prohlížeč

Doporučené zdroje informací

Accessible Rich Internet Applications (WAI-ARIA) 1.0. World Wide Web Consortium (W3C) [online]. 2014 [cit. 2015-06-24]. Dostupné z: http://www.w3.org/TR/wai-aria/

CONNOR, Joshue O. Pro HTML5 accessibility: building an inclusive web. New York: Distributed to the book trade worldwide by Springer Science Business Media, 2012, xix, 365 p. ISBN 978-1430241942.

LAWSON, Bruce, Remy SHARP a Bruce LAWSON. Introducing HTML 5. 2nd ed. Berkeley, CA: New Riders, 2012, xvi, 295 p. Voices that matter. ISBN 03-217-8442-1.

Notes on Using ARIA in HTML. World Wide Web Consortium (W3C) [online]. 2015 [cit. 2015-06-29]. Dostupné z: http://www.w3.org/TR/aria-in-html/

STEVENS, Luke a RJ OWEN. The truth about HTML5. New York: Apress, 2013. ISBN 978-1-4302-6415-6.

Předběžný termín obhajoby

2015/16 LS – PEF

Vedoucí práce

Ing. Petr Benda, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 28. 10. 2015

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 10. 11. 2015

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 06. 03. 2016

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "HTML5 - struktura a sémantika" jsem vypracoval samostatně pod vedením Ing. Petra Bendy, Ph.D. a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 6.3.2016 _____

Poděkování

Rád bych poděkoval Ing. Petru Bendovi, Ph.D. za jeho čas, cenné rady, připomínky a metodické vedení při vypracování práce.

HTML5 – struktura a sémantika

HTML5 – structure and semantics

Souhrn

Bakalářská práce se zabývá značkovacím jazykem HTML5 a nástrojem WAI-ARIA. Jejím úkolem je poukázat na sémantiku, kterou obě technologie poskytují.

Teoretická část je v první řadě zaměřena na vývoj HTML5, jeho strukturu a novou sémantiku. Dále jsou uvedeny a vysvětleny pojmy přístupnost a asistivní technologie, které úzce souvisí s touto problematikou. Poslední část obsahuje obecné informace o nástroji WAI-ARIA, jeho použití a vlastnosti.

Praktická část se zabývá využitím nových elementů HTML5 a atributů WAI-ARIA na webových stránkách. Testování použitelnosti je prováděno pomocí výstupů z asistivní technologie, použité u webových prohlížečů.

Summary

Bachelor thesis deals with markup language HTML5 and WAI-ARIA tool. The task of this work is point at semantics which provide both of these technologies.

The theoretical part of thesis is primaly focused on development HTML5, his structure and new semantics. The following are listed and explained concepts of accessibility and assistive technology which releated to this topic. Last part contains general information about WAI-ARIA tool, his usability and main properties.

The practical part of thesis deals with the use of new HTML5 elements and WAI-ARIA atributes on websites. Testing usability is based on outputs from assistive technology and web browsers.

Klíčová slova: html5, wai-aria, sémantika, přístupnost, web, standard, prohlížeč

Keywords: html5, wai-aria, semantics, accessibility, web, standard, browser

Obsah

1 Úvod

Jazyk HTML (HyperText Markup Language) je prostředek pro tvorbu internetových stránek a aplikací, bez kterého se v dnešní době při jejich vývoji neobejdeme. Během let prošel jazyk několika změnami, které ho pozitivně i negativně ovlivnily.

Aktuální verze s označením HTML5 je považována za webový standard, který oproti předchozí verzi HTML4.01 poskytuje řadu nových elementů a vlastností, které usnadňují vývoj a zlepšují přístupnost. Přestože je pokrok oproti předchůdci znatelný, z hlediska sémantiky a přístupnosti nepokrývá jazyk zdaleka všechno. Z toho důvodu existují podpůrné technologie, které umožňují nedostatky HTML vyřešit, nebo alespoň částečně zlepšit. Jednou z nich je WAI-ARIA, jde o soubor metadat umožňující přidat dodatečnou sémantiku tam, kde samotné HTML nestačí.

Tématem práce je rozlišení, kdy a v jakých situacích stačí využít nativní HTML, především nové elementy HTML5, a kdy je vhodné doplnit kód o sémantiku WAI-ARIA.

2 Cíle práce a metodika

2.1 Cíle práce

Cílem této bakalářské práce je poukázat na správné použití a výhody, které přináší nová sémantika v HTML5. Práce bude zaměřená na nové elementy, jejich podrobné vysvětlení a využití v praxi. Důraz bude kladen především na vzájemné propojení HTML5 elementů s technologií WAI-ARIA.

2.2 Metodika

Řešení problematiky bakalářské práce bude založeno na studiu a analýze odborných informačních zdrojů.

Teoretická část bude věnována především vysvětlení jednotlivých pojmů, správné HTML5 syntaxi a struktuře.

Praktická část bude důkladně analyzovat sémantiku vytvořené webové stránky. Tato webová stránka bude vytvořena celkem třikrát. První varianta bude obsahovat výhradně elementy HTML5, druhá varianta bude tvořena pomocí technologie WAI-ARIA a třetí varianta bude kombinovat výhody obou technologií. Součástí praktické části bude také podrobná analýza podpory použitých elementů a vlastností webovými prohlížeči. Výsledky práce budou přehledně vyhodnoceny a interpretovány.

3 Vývoj a přínos jazyka HTML 5

3.1 Historie

Historie jazyka HTML sahá do roku 1989 a 1990, kdy Tim Berners-Lee sepsal první specifikaci HTML a dal tak do budoucna základ nejpoužívanějšímu jazyku pro tvorbu webových stránek. V roce 1996 pomohl zformovat konsorcium W3C, které převzalo kontrolu nad dalším vývojem tohoto jazyka.[1]

Klíčovým rokem pro tento jazyk se stal rok 1998, kdy se W3C rozhodlo pozastavit vývoj HTML. Ve W3C věřili, že budoucností webových stránek bude XML, a tak po vydání poslední verze HTML 4.01 v roce 1999 přišli s novou specifikací XHTML 1.0. Ta měla postavený základ na jazyce XML a přejímala tedy i její pravidla o syntaxi, např. uzavírání všech značek. Krátce po vydání XHTML verze 1.0 začala pracovní skupina směřovat svůj vývoj na zcela novou verzi XHTML 2.0. Mělo se jednat o zcela nový jazyk, který ovšem nebude zpětně kompatibilní s předchozími verzemi HTML a XHTML.[2] S tím nebyli spokojeni vývojáři prohlížečů a většina webových vývojářů. Např. Bruce Lawson (pracovník firmy Opera a autor knihy Introducing HTML 5) napsal o XHTML 2.0:

„XHTML 2 was a beautiful specification of philosophical purity that had absolutely no resemblance to the real world.“ [1]

Volný překlad: *„XHTML 2 byla ztělesněním filozofické čistoty, která neměla žádnou podobnost s reálným světem.“*

HTML5 začalo jako reakce na kroky W3C a jejich vývoj XHTML 2.0. V roce 2004 dospěla skupina vývojářů ze společnosti Opera, Mozilla a Apple k názoru, že tvorba webových stránek už dávno není jenom o textu a obrázcích, ale že se čím dál více stávají atraktivní webové aplikace. V témže roce tato skupina představila konsorciu W3C svůj vlastní návrh specifikace zaměřený na webové aplikace, s tím ale v konsorciu nechtěli mít nic společného a tak tento návrh odmítli. Po tomto nezdaru zformovali reprezentanti Opery, Mozilly a Applu novou skupinu s názvem WHATWG (Web Hypertext Applications Technology Working Group). Tato nově vzniklá skupina brzy vydala svou vlastní specifikaci s názvem Web Applications 1.0, která byla zaměřena na psaní webových aplikací a obsahovala novou verzi HTML – HTML5. Roku 2007 si ve W3C

uvědomili svou chybu a požádali WHATWG o vzájemnou spolupráci. Vytvořili vlastní pracovní skupinu, která se připojila k WHATWG na vývoji již regulérního HTML5.[1] V roce 2011 se obě skupiny rozdělují. WHATWG o rok později zveřejňuje svoji verzi specifikace HTML - Living Standard, natož W3C v roce 2014 (28. Října) vydává HTML5 jako doporučený webový standard.[3]

3.2 Současný stav jazyka

Existují tedy dvě aktuální specifikace pro jazyk HTML5, HTML5 pod záštitou W3C a Living Standard spravovaný skupinou WHATWG. V úvodu obou specifikací se dočtete, že se jedná víceméně o totéž. WHATWG to ve své specifikaci stručně vysvětluje takto:

„The term "HTML5" is widely used as a buzzword to refer to modern Web technologies, many of which (though by no means all) are developed at the WHATWG. This document is one such; others are available from the WHATWG specification index.“[4]

Volný překlad: *„Termín HTML5 je obecně používán ve spojení s moderními technologiemi, z nichž mnohé (ovšem zdaleka ne všechny) jsou vypracovány na WHATWG. Tento dokument je jedním z nich, ostatní jsou k dispozici na WHATWG specification index.“*

Přestože se v obou případech jedná o specifikaci „HTML5“, existuje mezi oběma specifikacemi řada rozdílů. Bruce Lawson na svém blogu napsal, kterou specifikací by se vývojář měl řídit a proč:

„If you want to see what's already implemented in browsers now, look at W3C spec. If you want to see what might be coming (or how things may change) look at WHATWG spec.“[5]

Volný překlad: *„Pokud chcete vědět, co je již implementováno v prohlížečích, podívejte se na specifikaci W3C. Pokud chcete vědět, co by mohlo přijít nebo jak se věci mohou změnit, přečtěte si specifikaci od WHATWG.“*

3.3 Přínos HTML5

HTML4 a XHTML byly prosté značkovací jazyky, oproti tomu nové HTML5 poskytuje daleko více než práci s textem. Jeho součástí je řada API (Application Programming

Interface), u kterých nemusí být na první dojem znát, že jsou jeho součástí. HTML5 lze díky novým technologiím a vlastnostem rozdělit do následujících skupin:[6]

- **Vylepšená sémantika** – umožňuje díky novým sémantickým elementům lépe popsat obsah webových stránek, a tak např. asistivním technologiím, jako je screen reader, ulehčit práci s jeho čtením. Tento bod je stěžejní pro tuto práci.
- Konektivita – nabízí nové způsoby, jak se pomocí HTML5 připojit k serveru, např. technologie Web Socket, která vytvoří trvalé spojení mezi stránkou (klientem) a serverem.
- Offline úložiště – před nástupem HTML5 bylo pro webovou aplikaci zapotřebí neustálé připojení k internetu. Nyní je možné ukládat data na straně klienta, a tak zefektivnit práci v offline režimu.
- Multimedia – umožňuje manipulaci s multimediálním obsahem skrze nové elementy `audio` a `video`.
- 2D/3D grafika – připojení vektorové grafiky SVG a složitější vykreslování pomocí elementu Canvas.
- Výkon a integrace – zajistí efektivnější využití hardwaru
- Přístup k zařízení – umožní nové způsoby komunikace s uživatelem, např. pomocí „geolokačního“ API, které je schopno požádat o informaci o vaší poloze bez nutnosti jakýchkoliv doplňků.
- Pokročilé styly – nová řada kaskádových stylů CSS3, která umožňuje práci s animacemi, stínováním, přechody, atd.[6]

4 Syntaxe a nové elementy

4.1 Syntaxe

4.1.1 Deklarace typu dokumentu – DOCTYPE

Důležitou změnou v syntaxi je deklarace typu dokumentu tzv. DOCTYPE. Ten prošel radikální změnou – zkrátil se jeho zápis a změnil význam. Deklarace v předešlých verzích jazyka HTML byla dlouhá a špatně se pamatovala, jelikož bylo zapotřebí definovat odkaz na DTD (Document Type Definition). U HTML 5 není zapotřebí nic definovat, DOCTYPE zde slouží výhradně pro nastavení standardního vykreslovacího režimu.[7]

Přehled jednotlivých deklarací DOCTYPE:

HTML 4.01 Strict

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

HTML 4.01 Transitional

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

XHTML 1.0 Strict

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

XHTML 1.0 Transitional

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

HTML 5

```
<!DOCTYPE html>
```

4.1.2 Znaková sada a definování jazyka

Definování znakové sady v HTML se provádí pomocí elementu `meta`. Ten slouží obecně k definování různých metadat, jako jsou např. klíčová slova, popis stránky nebo jméno autora. Zápis znakové sady v HTML 4 vypadal takto:[1]

```
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
```

Oproti tomu zápis v HTML 5 je mnohem kratší a obsahuje pouze atribut `charset`:

```
<meta charset="UTF-8">
```

K definování jazyka slouží atribut `lang`. Oproti verzi HTML 4.01 se jedná o globální atribut, lze jej tedy použít v jakémkoliv HTML elementu. Slouží ke specifikaci jazyka u daného elementu. V případě definování jazyka pro celý dokument by jeho zápis vypadal takto:[1]

```
<html lang="cs">
```

4.1.3 Elementy link a script

Pokud jde o značky `link` a `script`, které se nejčastěji používají pro připojení kaskádových stylů a JavaScriptu, u nich není zapotřebí uvádět atribut „`type`“. Ten specifikuje typ media tzv. „MIME type“, v případě JavaScriptu šlo o hodnotu `text/javascript`. V HTML5 je tato hodnota nastavena jako výchozí, obdobně je tomu i v případě kaskádových stylů.[1]

4.2 Sémantické elementy

Sémantika je nauka o významu slov. V případě HTML jsou slova jednotlivé elementy. Obecně platí, že mezi sémantické elementy patří takové, které mají určitou vypovídací hodnotu o svém obsahu. Mezi ně patří např. element `table`. Na první pohled je jasné, že definuje tabulku. Ovšem ne vždy je důležité definovat obsah uvnitř elementu, např. pokud jde čistě o stylizační prvek, typicky element `div`, který může do jisté míry za vytvoření nových elementů.

Před příchodem HTML5 se struktura webové stránky tvořila právě pomocí elementů `div`, které ovšem nenesou žádný sémantický význam, a tak bylo pro prohlížeče obtížné rozpoznat jednotlivé úseky stránky. Většina stránek je obvykle tvořena částmi, jako je hlavička, navigace, postraní sloupce, hlavní obsah a patička. V HTML 4 neexistoval žádný element, který by jasně definoval tyto části, a tak se celá struktura skládala z elementů `div`. Proto se podle nejpoužívanějších názvů tříd a id vytvořily nové elementy.[2]

Mezi nejzákladnější patří:

- Article
- Section
- Aside
- Header
- Footer
- Nav
- Main
- Figure a figcaption
- Details a summary
- Progress a meter
- Time
- Video a audio

Article
Element `article` představuje nezávislý, samostatný obsah, který je z dokumentu stránky možné libovolně vyjmout a číst jej odděleně, aniž by se změnil jeho význam. Obvykle se jedná o různé články, novinky nebo komentáře. Každý `article` by měl obsahovat nadpis (elementy h1-h6), který jej identifikuje. Složitější (případně delší) článek může obsahovat ostatní strukturální elementy jako hlavičku s patičkou, vnořené sekce a další články nebo navigaci.[9]

Section
Element `section` svým způsobem velmi připomíná `article` a jejich význam se často zaměňuje. Zatímco obsah uvnitř elementu `article` má význam sám o sobě a lze jej z dokumentu libovolně vyjmout a použít jej jinde, obsah zabalený v elementu `section` je závislý na zbytku dokumentu. Dobrým příkladem sekcí jsou např. kapitoly, panely v dialogovém okně nebo očíslované oddíly v diplomové práci.[9]

Aside
Element `aside` slouží k vyznačení dodatečného obsahu určitého úseku dokumentu. S tímto úsekem souvisí, ale není pro jeho obsah nezbytný. Obvykle se jedná o dodatečné informace, které např. vysvětlují určitý pojem.[9]

Header
Element `header` slouží k vyznačení hlavičky určité sekce nebo celé stránky. Obvykle obsahuje elementy pro nadpis společně s dalším obsahem, jímž mohou být úvodní nebo navigační prvky v případě článku nebo hlavní logo pro celou stránku.[9]

Footer
Element `footer` definuje patičku celé stránky nebo dané části. Měl by obsahovat informace o sekci, ke které se vztahuje, jako autora, kontaktní informace, autorská práva, odkazy na související dokumenty atp. Patička se nemusí vyskytovat nutně na konci sekce, jak tomu často bývá. Jak element `footer` tak `header` se mohou v dokumentu vyskytovat vícekrát.[9]

Nav
`Nav` je element obsahující skupinu odkazů, které odkazují na části dokumentu nebo na jiné stránky. Není ovšem vhodné každou skupinu odkazů, která se v dokumentu vyskytuje, vnořit do elementů `nav`, tento element slouží výhradně pro hlavní navigace. Pod pojmem „hlavní navigace“ se myslí navigace např. celého dokumentu nebo uvnitř článku, kde může sloužit jako obsah.[10]

Main
Element `main` vymezuje hlavní část dokumentu. Důvodem, proč byl tento element přidán, bylo vytvořit HTML element, který by byl alternativou za roli main ze specifikace WAI-ARIA (viz. Úvod do WAI-ARIA). Pomohl by tak, asistivním technologiím, jako je screen reader, pochopit, kde začíná hlavní obsah dokumentu. V celém dokumentu se smí nacházet právě jeden element `main` a nesmí být potomkem žádného z elementů `article`, `aside`, `footer`, `header` nebo `nav`.[11]

Figure a figcaption
Element `figure` doplňuje hlavní obsah (např. článek) o nějaký samostatný objekt, který je pro tento obsah nezbytný. Objekt je z hlavního obsahu odkazován, nezáleží na jeho pozici, a proto může být přesunut do jiného bloku, aniž by se tím narušil tok dokumentu. Nejčastěji jde o obrázek, graf a ilustraci, ale může se jednat i o tabulku nebo kus zdrojového kódu.

`Figcaption` má funkci legendy (popisku), jeho výskyt není povinný, ale doporučuje se.[12]

Details a summary
Element `details` se používá k označení obsahu, který může uživatel skrýt nebo odkrýt. Starší verze HTML neobsahovaly žádný element, jenž by dovedl něco podobného, a proto se tato funkce nejčastěji realizovala pomocí JavaScriptu.

Pokud webový dokument obsahuje element `details`, prohlížeč zobrazí pouze jeho titulek, který se nastavuje pomocí elementu `summary`, po rozkliknutí se zobrazí skrytý obsah. Součástí je i speciální atribut „`open`“, který pokud je nastaven, zobrazí sekci jako otevřenou.[13]

Progress a meter
Element `progress` poskytuje snadnou cestu, jak vytvořit progressbar (ukazatel průběhu). Ten se hodí pro spoustu webových aplikací, v nichž je dobré zobrazit aktuální stav vykonávané operace, např. při nahrávání souboru (uživatel vidí aktuální stav nahrávání).

`Meter` je svým způsobem velmi podobný elementu `progress`, také zobrazuje aktuální stav, ale za jiným účelem. Zatímco `progress` zobrazuje aktuální stav dané operace, meter se obvykle využívá jako měřič (např. volného místa na disku).[2]

Time
`Time` je element reprezentující čas v 24hodinovém formátu nebo datum podle gregoriánského kalendáře. Smyslem elementu je udat přesný časový údaj v univerzálním tvaru, aby byl strojově čitelný. To má za následek např. přesnější výsledky hledání, které mohou vyhledávače poskytnout. Časový údaj lze jednoduše zadat např. takto:

```
<time>20:00</time>
```

Nicméně tento zápis by byl velmi nepraktický, jelikož údaj uvnitř elementu musí zachovat určitý formát. Řešením je speciální atribut „`datetime`“. Ten umožňuje libovolnému textu přiřadit časový údaj.[14]

Např. u věty „Zítra mám narozeniny.“ by její zápis vypadal takto:

```
<p><time datetime="2015-10-27">Zítra</time> mám narozeniny.</p>
```

Video a audio
Jednou z velkých výhod HTML5 je schopnost přehrát v prohlížeči video a audio bez nutnosti instalace doplňku třetích stran např. Flash, QuickTime, atp. Video (nebo audio) se

vkládá skrze atribut `source`, který obsahuje adresu k souboru videa. Druhou možností je využití elementu `source`, s jehož pomocí lze vložit několik video souborů (každý v jiném formátu). Prohlížeče pak zvolí první podporovaný formát. Mezi podporované formáty patří:

- video: MP4, WebM, Ogg
- audio: MP3, Wav, Ogg

Další důležitou vlastností elementů `video` a `audio` je atribut `controls`. Pokud je atribut nastaven, zobrazí prohlížeč u videa a audia kontrolní panel pro jejich ovládání.[7]

4.3 Formulářové prvky

Formuláře jsou nedílnou součástí webových stránek, jedná se o jedinečné prvky, které slouží k základní interakci mezi uživatelem a webovou stránkou. S využitím formulářů, uživatelé vyplňují různé dotazníky a registrace, přihlašují se na e-mail nebo vyhledávají informace. HTML5 poskytuje řadu nových elementů, atributů a celkově nové vlastnosti, jak ulehčit vývojářům práci při jejich vytváření.[15]

4.3.1 Nové elementy

Datalist

Element datalist reprezentuje seznam předdefinovaných hodnot (možností), které jsou poté navrhovány v elementu input. Datalist funguje podobně jako vyhledávací pole v prohlížeči, které nabízí různé možnosti při psaní.[7]

Output

Element `Output` představuje výsledek nějaké početní operace, nejčastěji se jedná o výstup JavaScriptu. Obvykle tento element zahrnuje atribut `for`, jenž obsahuje jména elementů, kterých se výsledek týká (atribut není povinný).[7]

4.3.2 Nové typy inputů

Inputy jsou vstupní pole, která se využívají při tvorbě formulářů a která umožňují uživateli vkládat data. Jejich podobu a využití určuje atribut „`type`“. HTML5 přináší několik

nových typů polí, které vývojářům krátí čas při tvorbě formulářů a uživatelům ulehčuje práci v jejich používání.[15]

Tel, url a email

U těchto tří typů jde v podstatě o standardní textové pole, jehož vstup musí odpovídat určité podmínce (vzoru). V případě zápisu `<input type="email">` musí vstup odpovídat emailové adrese. Není proto zapotřebí využívat JavaScript, který by byl v případě použití prostého textového pole nezbytný pro vytvoření podmínky.[15]

Vyhledávací pole

Vyhledávací pole funguje opět stejně, jako to textové, ovšem s tím rozdílem, že při psaní se uživateli zobrazí malý křížek. Pokud uživatel na křížek klikne, obsah pole se smaže. K vytvoření vyhledávacího pole slouží zápis <input type="search">.[15]

Typy polí pro vložení data a času

Pokud jde o vkládání data a času, existuje celkem pět možností, jak uživateli poskytnout pole pro jejich vložení. Jejich zápis vypadá následovně:

- `<input type="date">` - uživatel definuje datum (rok/měsíc/den)
- `<input type="month">` - slouží k výběru měsíce a roku
- `<input type="week">` - slouží k výběru týdne a roku
- `<input type="time">` - slouží k zápisu času
- `<input type="datetime-local">` - uživatel vybírá datum a čas [15]

Vkládání číselné hodnoty

Pro vkládání nebo nastavení číselné hodnoty je nově možné využít pole `number` a `range`. Pokud jde o `number`, uživatel zadává přesnou hodnotu, kterou lze omezit atributy `min` a `max`.

V případě typu `range` nastavuje uživatel hodnotu pomocí slideru (posuvníku). Opět lze omezit vstup od uživatele atributy `min` a `max`.[15]

Výběr barvy

Inputem `type="color"` lze vybrat barvu a vrátit její hodnotu v hexadecimálním zápise. Uživatel vybírá barvu skrze color picker, který je součástí prohlížeče.[15]

5 Přístupnost

5.1 Obecně o přístupnosti

Webová přístupnost (Web Accessibility) je obecně chápána jako bezbariérovost v oblasti internetu. Jejím úkolem je odstranit překážky, které brání přístupu osobám s těžkým zdravotním postižením. Služby a informace, které web obsahuje, by měly být dostupné všem, aby i lidé se zdravotním postižením měli stejnou možnost přístupu a mohli tak bez sebevětších problémů využívat všechny služby a funkce, které daný web nabízí.[16]

Existuje celá řada definic, kterými lze popsat webovou přístupnost. Jednou z nich je definice od W3C, přesněji z jednoho jeho odvětví – Web Accessibility Initiative (WAI). Tato skupina se již několik let snaží zpřístupnit web lidem se zdravotním postižením vydáváním různých metodik a postupů. Ve svém článku „Introduction to Web Accessibility“ popisují přístupnost takto:

„Web accessibility means that people with disabilities can use the Web. More specifically, Web accessibility means that people with disabilities can perceive, understand, navigate, and interact with the Web, and that they can contribute to the Web. Web accessibility also benefits others, including older people with changing abilities due to aging.“[17]

Přístupnost ovlivňuje celá řada aspektů - zdravotní postižení uživatele, technické zpracování webu, zkušenosti konkrétního uživatele s prací s webem, zkušenosti konkrétního uživatele s prací s asistivní technologií, použitá asistivní technologie, její konfigurace a verze, použitý prohlížeč a preference a zvyklosti konkrétního uživatele. Při tvorbě přístupného webu je tedy vhodné věnovat pozornost všem aspektům, které přístupnost ovlivňují. Jen tak lze docílit toho, že web bude přístupný reálně, a ne jen formálně.[18]

Při realizaci webové prezentace si je důležité uvědomit, že ji v budoucnu může navštěvovat široké spektrum uživatelů. Každý uživatel je odlišný. Ať už má zdravotní postižení nebo ne, tak navštěvuje webové portály z různých prohlížečů a zařízení. Aby byl web přístupný, je nutné zajistit, aby jej každý uživatel bez rozdílu v tom, odkud přichází, jak je technický zdatný, zdali má nějaký druh zdravotního postižení nebo jaký používá prohlížeč, mohl číst a využívat všech jeho služeb.[16]

Přístupnost se úzce překrývá s ostatními praktikami při tvorbě webových prezentací a aplikací, jako je mobilní web design, nezávislost zařízení, multi-modální interakce, použitelnost a design pro starší uživatele nebo optimalizace pro vyhledávače SEO (Search Engine Optimization). Případové studie taktéž ukazují, že přístupné weby mají mnohem lepší výsledky vyhledávání než webové stránky, které mají s přístupností problémy. Z toho plyne větší návštěvnost. Přístupnost dále snižuje náklady na údržbu, a je proto zásadní pro organizace a vývojáře, kteří tvoří webové prezentace a aplikace na profesionální úrovni.[19]

5.2 Asistivní technologie

S přístupností se pojí termín „asistivní technologie“ a způsobů, jak tento pojem definovat je řada. Ve výstupech projektu ATIS4all (Assistive Technologies and Inclusive Solutions for All), kterého se v letech 2011 až 2013 účastnilo Ministerstvo vnitra České republiky, jsou asistivní technologie definovány jako:

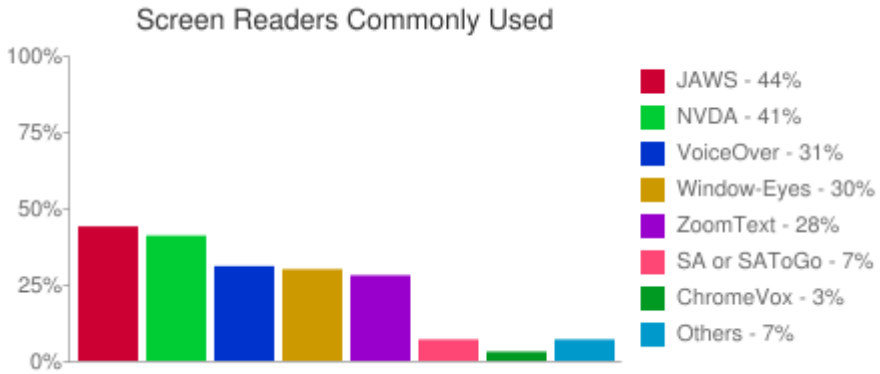
„jakýkoliv nástroj, zařízení, software nebo systém, využívající zpravidla moderní technologie (zejména senzory, aktuátory, informační a komunikační technologie) s cílem posílit, udržet nebo zlepšit funkční schopnosti jedinců se speciálními potřebami, a tím jim usnadnit každodenní život a zlepšit kvalitu jejich života, samostatnost a soběstačnost“.[20]

Za jedince se speciálními potřebami jsou zde považováni senioři, zdravotně postižení a chronicky nemocní lidé. Tyto lidi lze nazvat jako lidi s hendikepem a úkolem asistivních technologií je pomoci tento hendikep překonat.[20]

5.2.1 Screen readers

Odečítače obrazovky (screen readers) jsou jedním typem asistivní technologie, převážně jsou využívány nevidomými a osobami, které trpí zrakovým postižením. Jde o speciální software na bázi text-to-speech, jenž převede obsah obrazovky do syntetizované řeči. K interakci uživatel nejčastěji využívá klávesnici, kdy se pomocí klávesových zkratek pohybuje po obsahu a vykonává určité akce. V rámci operačního systému fungují odečítače velmi dobře a jsou tak schopny hlasově interpretovat veškerou práci uživatele. Problém nastává u webových aplikací a stránek, které často nedbají na přístupnost a odečítače tak mají potíže s jejich interpretací.[7]

Webaim (portál zabývající se webovou přístupností) dělá od roku 2009 průzkumy, které mapují oblíbenost jednotlivých softwarů. Podle posledního průzkumu z léta roku 2015, patří mezi tři nejpoužívanější odečítače JAWS, následovaný těsně NVDA a třetí místo náleží VoiceOver. Více znázorňuje následující graf:[8]



Obrázek 1 - Běžně používané odečítače obrazovky za rok 2015.[8]

6 WAI-ARIA

6.1 Úvod do WAI-ARIA

Zkratka WAI-ARIA (dále už jen ARIA) ukrývá spojení „Web Accessibility Initiative - Accessible Rich Internet Applications“ a jde o webovou specifikaci od W3C. Jejím účelem je pomoci zlepšit sémantiku webových stránek a aplikací, a zajistit tak asistivním technologiím (např. screen readers) informace, které jim standardní HTML poskytnout nedokáže. Zároveň nemá žádný vliv na to, jak jsou HTML elementy zobrazeny nebo jaké je jejich chování v prohlížečích. ARIA je pouze jakási „popisná vrstva“, sloužící k předání dodatečných informací asistivním technologiím. K poskytnutí těchto informací slouží ARIA atributy, ty jsou rozděleny do kategorií na role, stavy a vlastnosti.[21]

6.1.1 Role

Atribut role slouží k popsání elementu, co představuje nebo co dělá. Většina HTML elementů má již přednastavenou roli, např. element pro tlačítko – button má přednastavenou roli „button“. S ARIA lze elementům přiřadit role, které budou lépe definovat jejich podstatu a chování. Tyto role lze využít dvěma způsoby. Zaprvé lze s jejich pomocí popsat širokou škálu interaktivních prvků (widgets), jako jsou např. menu, listbox nebo slider. Zadruhé je možné přiřadit role jednotlivým částem stránky a lépe tak popsat strukturu celého dokumentu.[22]

Rozdělení rolí do kategorií:

- Abstract Roles - slouží k definování podstaty rolí (neměly by být používány)
- Widget Roles - definují vlastní komponenty (widgets)
- Structure Roles - popisují strukturu a organizaci obsahu stránky
- Landmark Roles - slouží k definování jednotlivých oblastí stránky (viz. Aria Landmarks, jejich význam a použití)[22]

6.1.2 Stavy a vlastnosti

Mezi další atributy, které ARIA skrze HTML poskytuje, patří „Stavy a vlastnosti“. Ty bývají nejčastěji používány k podpoře rolí, které se v dokumentu vyskytují. Vlastnosti obvykle popisují vztah s jinými elementy a z větší části se nemění, jakmile jsou nastaveny. Příkladem je `<input aria-required="true">`, tato vlastnost sdělí asistivní technologii,

aby na input nahlížela jako na povinný (uživatel jej musí vyplnit). Stavy jsou více dynamické vlastnosti, které definují aktuální stav, ve kterém se element nachází. Nejčastěji bývá jejich hodnota aktualizována JavaScriptem. Příkladem stavu je `<input aria-disabled="true">`, tato vlastnost nastaví input jako „zakázaný“, nicméně jeho hodnota může být snadno změněna na základě uživatelského vstupu. Atributy pro nastavení stavů a vlastností vždy začínají prefixem „`aria-`“.[22]

6.1.3 Live Regions

Z hlediska přístupnosti nastává jeden velký problém, jakmile se obsah v rámci webové stránky začne dynamicky měnit. V takovém případě, pokud na stránce proběhne změna, mají asistivní technologie dvě možnosti - buď přečíst celý obsah znovu, nebo změnu obsahu zcela ignorovat. V dnešní době, kdy má dynamický měnící se obsah každá druhá stránka (v lepším případě), by uživatel využívající asistivní technologii přišel o důležité informace. S ARIA může vývojář identifikovat oblasti, které se mají dynamicky měnit jako tzv. „Live Region“ (živá oblast). Živá oblast umožňuje aktualizaci obsahu způsobem, kterému asistivní technologie rozumí a dokážou se s ním vypořádat.

Pro vytvoření živé oblasti slouží atribut `aria-live`, který se nastaví elementu, jenž se bude dynamicky měnit. Podle hodnoty `off`, `polite` a `assertive` se určí, jak se asistivní technologie zachová, když se obsah elementu změní.

V případě hodnoty `off` (`aria-live="off"`) asistivní technologie neoznámí změnu. Toho lze využít u nedůležitých změn.

U hodnoty `polite` bude uživatel informován o změně obsahu, jakmile dokončí aktuální úlohu, pak může ihned přejít na aktualizovaný obsah. Tato hodnota se nejčastěji využívá při aktualizaci obsahu, jako je např. chat.

Pokud je hodnota nastavena na `assertive`, uživatel je upozorněn na změnu obsahu okamžitě. Tato hodnota bývá nejčastěji použita pro chybové zprávy.

Za pomoci živých oblastí lze také definovat obsah, který bude přečten, když dojde ke změně.[22]

6.1.4 Přístup z klávesnice

Navigace skrze klávesnici je z hlediska přístupnosti klíčová, za času HTML4 bylo možné získat fokus skrze klávesnici (pomocí klávesy TAB) pouze na odkazy nebo formulářové prvky. Pohyb po stránce pouze za pomocí klávesnice, byl proto velmi omezený, a pokud stránka obsahovala složitější prvky, např. drop-down (rolovací) menu, uživatel klávesnice nemohl jejich funkce využívat. ARIA se tento problém snažila vyřešit rozšířením atributu `tabindex` tak, že jej bylo možné nastavit všem elementům, ty pak získaly fokus z klávesnice. Dnes je tato vlastnost (atribut) součástí specifikace HTML5.[23]

6.1.5 Základní pravidla pro používání ARIA v HTML

Specifikace ARIA obsahuje celkem pět pravidel, kdy je vhodné použit její atributy (role, stavy a vlastnosti) v HTML, jak je použít a čeho se vyvarovat. Tyto pravidla nejsou zavazující a jejich nedodržení nemusí vždy vyústit v chybu.[24]

První pravidlo

V případě prvního pravidla se ke zlepšení sémantiky a přístupnosti doporučuje využívat především nativní elementy HTML a po atributech ARIA sáhnout teprve tehdy, kdy je to nezbytné a samotné HTML nestačí. ARIA se doporučuje využít v následujících případech:

- Potřebná funkce v HTML není zastoupena, HTML tedy neobsahuje nativní element, který by tuto funkci mohl pokrýt, nebo tento element obsahuje, ale jeho podpora není kompletní.
- Pokud vizuální omezení konstrukce vylučuje použití daného prvku HTML, jelikož tento element nemůže být stylizovaný, jak je požadováno.[24]

Druhé pravidlo

Doporučuje se neměnit sémantiku nativních elementů HTML pomocí rolí ARIA, pokud to není nezbytně nutné. Jakmile je HTML elementu přidána ARIA role, původní sémantický význam daného elementu se přepíše podle přidané role. Pokud je dodržováno první pravidlo, pak by tento problém neměl nikdy nastat, nicméně vždy nelze využít nativní elementy. Pro ilustraci následuje nevhodné použití atributu `role="button"` v elementu pro nadpis:

```
<h1 role="button">heading button</h1>
```[24]

Třetí pravidlo

Třetí pravidlo se zaměřuje na přístupnost z klávesnice. Jakýkoliv interaktivní prvek, který je ovládán pomocí klikání, funkce drag and drop, scrollování, atp., musí uživatelům taktéž umožnit přístup a ovládání skrze klávesnici.[24]

Čtvrté pravidlo

Čtvrté pravidlo je velmi specifické, jelikož se týká dvou konkrétních ARIA atributů: `role="presenation"` a `aria-hidden="true"`. Není vhodné, aby byl jeden z těchto atributů použit na viditelný element, u kterého lze získat fokus např. na tlačítko.[24]

Páté pravidlo

Každý interaktivní prvek musí být pojmenován tak, aby měl platné tzv. „accessible name“, jde o jednu z vlastností Accessibility API. Pokud tato vlastnost nemá hodnotu, asistivní technologie nemohou s určitostí zjistit, o jaký interaktivní prvek se jedná, případně jaký je jeho účel. Pro ilustraci, následující kód má viditelný název, ale nemá platné „accessible name“ (input není propojený s názvem):

```
Název <input type="text">
```

nebo

```
<label>Název</label> <input type="text">
```

Naopak následující zápis již má platné „accessible name“ (input je propojený s názvem):

```
<label>Název<input type="text"></label>
```

nebo

```
<label for="name">Název</label> <input type="text" id="name">
```

nebo

```
<input type="text" id="name" aria-label="Název">
```

Více o pojmenování vlastních prvků a komponent v kapitole – 6.3 Popis elementů a vlastních komponent.[24]

6.2 Aria Landmarks, jejich význam a použití

Většina webových stránek má jeden společný rys – kostru. Tato kostra se skládá z určitých částí, které obsahuje skoro každý web, patří mezi ně záhlaví a zápatí stránky, navigace a hlavní obsah. K tomu, aby se uživatel na stránce mohl pohybovat, potřebuje tyto části identifikovat. Pro uživatele, který netrpí zrakovým postižením, není žádný problém se na webu zorientovat a rychle pochopit účel jednotlivých oblastí. Pokud jde o zrakově postiženého uživatele, je nutné ho nějak informovat o jednotlivých oblastech, k čemu slouží a v jaké se uživatel právě nachází. ARIA poskytuje relativně snadný způsob, jak tyto části popsat a následně je identifikovat pomocí asistivních technologií. Slouží k tomu sada specializovaných rolí, tyto role se nazývají „landmarky“. (7)

ARIA poskytuje celkem osm landmarků:

Application - představuje oblast stránky, která vykonává řadu úkolů pro své uživatele, případně se očekává, že se tato oblast bude chovat jako desktopová aplikace. Důrazně se doporučuje, aby se tato role používala pouze zřídka.

Banner - oblast stránky, která obsahuje záhlaví stránky. Většina obsahu uvnitř banneru je tzv. „site-oriented“, obvykle se jedná o logo webu, hlavní nadpis (titulek) stránky, případně o vyhledávací nástroje. Často se tato oblast nachází v horní části stránky.

Complementary - reprezentuje libovolnou sekci, která je spjatá s hlavním obsahem dokumentu. Tato sekce může být oddělena, aniž by ztratila svůj význam.

Contentinfo - oblast stránky, která obsahuje informace o autorských právech, odkazy na zásady ochrany osobních údajů a obecně zápatí celého dokumentu.

Form - oblast stránky, která obsahuje formulář a všechny jeho vnořené elementy.

Main - slouží k identifikaci hlavní části (obsahu) stránky. Obvykle následuje po sekci s navigací. Dokument by měl obsahovat právě jeden element s rolí main.

Navigation - oblast stránky obsahující hlavní navigaci, skládá se z kolekce odkazů. Dokument může obsahovat více oblastí s touto rolí.

Search - slouží k identifikaci vyhledávacích nástrojů, obvykle jde o vyhledávací formulář. Doporučuje se, aby byl tento landmark použit v sémanticky neutrálních elementech jako je `div` nebo přímo v elementu `form`.[25]

6.3 Popis elementů a vlastních komponent

V případě přístupnosti se weboví vývojáři a designéři často potýkají s problémem, jak popsat své vlastní komponenty nebo sekce tak, aby o nich asistivní technologie dokázali získat informace a ty mohly dále předat svým uživatelům.[7] Samotné HTML poskytuje řadu možností, jak jednotlivé prvky a vlastní komponenty popsat. Nejjednodušší, ale zároveň nejméně spolehlivou, možností je globální atribut `title`. Ten je možné přidat ke všem elementům HTML a vyjádřit tak o nich dodatečné informace. Při standardním prohlížení se popisek uvnitř atributu `title` zobrazí pomocí bubliny (tooltip). Mnohem zajímavější způsob, jak popsat elementy, nabízí atribut `alt`. Ten je ovšem omezen pouze pro určité elementy, nejčastěji je součástí elementu `img` (prvek pro vkládání obrázků). `Alt` slouží pro vložení alternativního textu, který se zobrazí v případě, že se obrázek z nějakého důvodu nenačte. V případě hlasových čteček, jde o text, který je přečten, jakmile se uživatel nachází v místě pole obrázku. ARIA nabízí atributy, které disponují podobnými vlastnostmi jako atribut `alt`, s tím rozdílem, že je lze použít u všech elementů.[26]

Aria-label

Prvním atributem je `aria-label`, jenž funguje obdobně jako atribut `alt`. Do atributu se vloží text symbolizující popisek daného elementu. V následující ukázce je tlačítko, které je zobrazeno pomocí křížku, zároveň obsahuje atribut `aria-label` s hodnotou „Zavřít“, která nahradí původní text (tedy křížek).

```
<button aria-label="Zavřít">×</button>
```

Screen reader by svého uživatele informoval, že se jedná o tlačítko „Zavřít“.[27]

Aria-labelledby

Občas se na stránce vyskytuje text, který by dokázal jednoznačně popsat určitý element. Místo toho, aby se text zkopíroval a vložil do `aria-label`, existuje další atribut s názvem

`aria-labelledby`, umožňující propojit daný element s obsahem jiného elementu přes id.[27]

Jako příklad poslouží navigační sekce:

```
<nav role="navigation" aria-labelledby="navheading">
  <h3 id="navheading">Hlavní menu</h3>
  …
</nav>
```

Aria-describedby

Vlastnost tohoto atributu je velmi podobná předešlému `aria-labelledby`, který slouží převážně k vyjádření názvu daného prvku, `aria-describedby` poskytuje dodatečné informace.[27]

```
<label for="policko>Název pole</label>
<input id="policko" aria-describedby="popisek">
<i id="popisek>Popisek políčka</i>
```

Aria-hidden

Jsou situace, kdy existuje obsah, v jehož případě nemá smysl, aby jej odečítač obrazovky četl, nebo není potřeba mu nastavovat jiný text přes `aria-label`. Pomocí atributu `aria-hidden` lze obsah skrýt. Takovým případem může být tlačítko tvořené symbolem i textem.[27]

```
<button><span aria-hidden="true">×</span>Zavřít</button>
```

7 ARIA a HTML5

7.1 Implicitní ARIA sémantika

V některých případech lze sémantiku HTML elementů vyjádřit pomocí rolí, stavů a vlastností, které poskytuje ARIA skrze své atributy. Pokud je možné HTML elementy takto vyjádřit, lze o nich prohlásit, že mají nastavenou výchozí ARIA sémantiku tzv. implicitní. V takovém případě platí první pravidlo použití ARIA v HTML, kdy k vyjádření sémantiky je vždy lepší použít nativní element, než vkládat ARIA atributy (viz. pravidla).[28]

„ARIA roles do not add anything to the default semantics of most HTML elements.“ (Steve Faulkner)[29]

Vkládáním nadbytečných ARIA atributů elementům, které již mají přednastavenou sémantiku, vzniká redundantní kód.

V HTML5 specifikaci, jejíž součásti je taktéž specifikace ARIA, se dále uvádí, že není nutné definovat ARIA role a atributy, které slouží k vyjádření implicitní ARIA sémantiky, jelikož o tuto starost se stará webový prohlížeč.[29]

„In the majority of cases setting an ARIA role and/or aria- attribute that matches the default implicit ARIA semantics is unnecessary and not recommended as these properties are already set by the browser.“ (Specifikace HTML5)* [29]

Několik příkladů, kdy vzniká redundantní kód:

Přidání implicitní role interaktivnímu elementu (např. tlačítko) je podle doporučení HTML5 považováno za redundantní kód:

```
<button role="button">press me</button>
```

Definování stavu nebo vlastnosti, pokud v HTML existuje alternativa:

```
<input type="text" required aria-required="true">
```

Přidání role a stavu nebo vlastnosti strukturálnímu elementu, který je již řadu let implementován:[28]

```
<h1 role="heading" aria-level="1">heading text</h1>
```

7.2 ARIA a nové HTML5 elementy

HTML5 přineslo řadu nových elementů, které zlepšují sémantiku a slouží k vyznačení jednotlivých sekcí stránky nebo aplikace. Tyto elementy obsahují implicitní sémantiku, která mapuje jednotlivé role ARIA:

Nativní element HTML5	**ARIA role**
article	role="article"
aside	role="complementary"
footer	role="contentinfo"
header	role="banner"
main	role="main"
nav	role="navigation"
section	role="region"

Tabulka 1 - Několik příkladů nativních elementů HTML5 s implicitní ARIA sémantikou (autor)

U footer a header to platí pouze tehdy, kdy tyto elementy nejsou potomky jednoho z elementů article nebo section.

Webový prohlížeč v těchto případech vystaví sémantiku jednotlivých elementů pomocí dané role, tato vlastnost ovšem musí být v prohlížečích správně implementována. Většina moderních prohlížečů takovou vlastnost podporuje, existují ale i výjimky.[28]

7.3 Nativně nepodporovaná ARIA

Jak bylo zmíněno výše, HTML5 poskytuje řadu elementů a vlastností s implicitní ARIA sémantikou, která je nastavena v prohlížečích. ARIA navíc obsahuje širokou škálu rolí, stavů a vlastností, které zatím v HTML5 nemají nativní podobu. Podle specifikace „Notes on Using ARIA in HTML" z 21. Května 2015 obsahuje ARIA celkem 29 rolí a 22 stavů a vlastností (aria-* atributy), které v HTML5 nemají nativní zastoupení.[30] Kompletní soupis těchto ARIA atributů se nachází ve specifikaci na této adrese: <https://www.w3.org/TR/aria-in-html/#aria-roles-and-properties-not-available-as-features-in-html>.

8 Praktická část

V teoretické části byla popsána sémantika, kterou HTML5 poskytuje prostřednictvím svých elementů a kterou ARIA umožňuje přidat do HTML kódu prostřednictvím svých atributů. Na základě těchto poznatků je v praktické části testováno, do jaké míry jsou nové elementy podporovány v předních webových prohlížečích a jaká je jejich použitelnost z hlediska přístupnosti. Dále je na několika statických příkladech ukázáno využití ARIA a také to, do jaké míry její použití ovlivňuje asistivní technologie zejména odečítače obrazovky. V závěru práce je pozornost věnována vlastní komponentě, která byla pro tyto účely zhotovena. Všechny testované stránky jsou součástí přílohy na konci dokumentu.

K testování a porovnávání jsou zapotřebí webové prohlížeče. W3C disponuje jednoduchou statistikou nejpoužívanějších prohlížečů za uplynulé roky. Podle této statistiky byly vybrány tři přední prohlížeče, které dosáhly za rok 2015 největší popularity. Mezi vybrané zástupce patří jejich nejnovější dostupné verze:

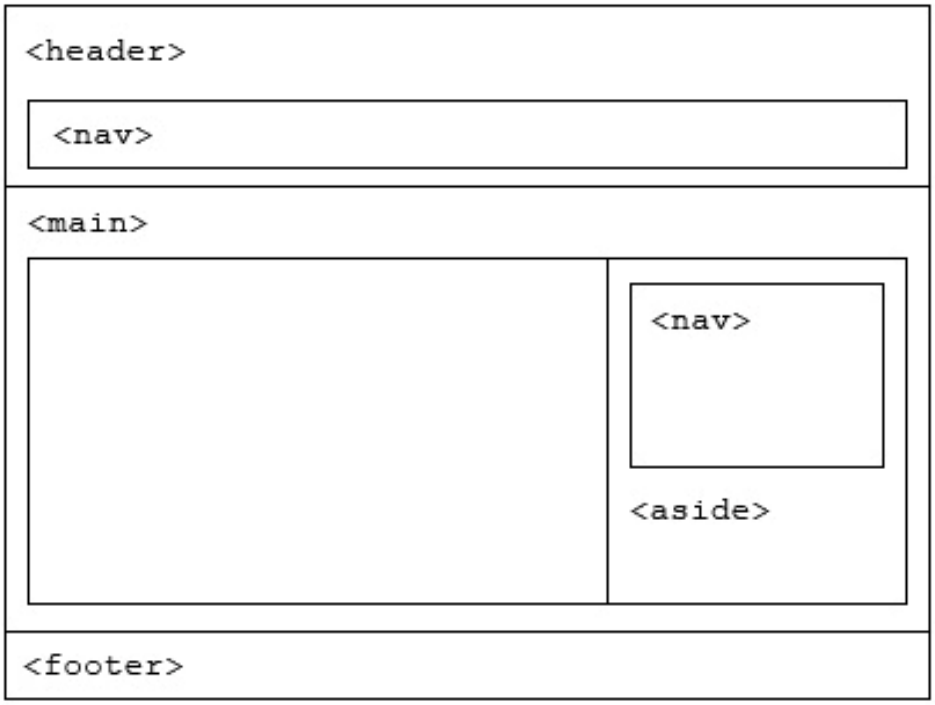
- Google Chrome 47
- Mozilla Firefox 43
- Internet Explorer 11

Kompletní statistika: <http://www.w3schools.com/browsers/browsers_stats.asp>

V případě asistivní technologie poslouží odečítač obrazovky NVDA ve verzi 2015.4. Aktuální verzi softwaru je možné pod licencí GPL (General Public License) stáhnout zdarma z portálu <http://www.nvaccess.org/>.Testování proběhlo na operačním systému Windows 7.

8.1 Použití HTML5

Aby mohlo začít testování jednotlivých elementů, bylo zapotřebí vytvořit webovou stránku. Pro tyto účely postačila jednoduchá kostra, složená ze základních oblastí, které obsahuje většina dnešních stránek. Stránka je rozdělena do následujících oblastí: Hlavička s nadpisem, hlavní navigace, vyhledávací panel, sekce s obsahem, postranní panel a kontaktní informace (patička stránky). Obrázek níže znázorňuje uspořádání oblastí na stránce a elementy, které byly k jejich vyznačení použity.



Obrázek 2 - Kostra stránky (autor)

Do takto připravené stránky se postupně začaly vkládat ostatní elementy, které byly zmíněny a vysvětleny v teoretické části. Jakmile byly zahrnuty všechny, celá stránka se otestovala napříč zvolenými prohlížeči. Výsledky, jak jednotlivé elementy a ostatní vlastnosti dopadly, obsahuje tabulka v příloze – Příloha 1.

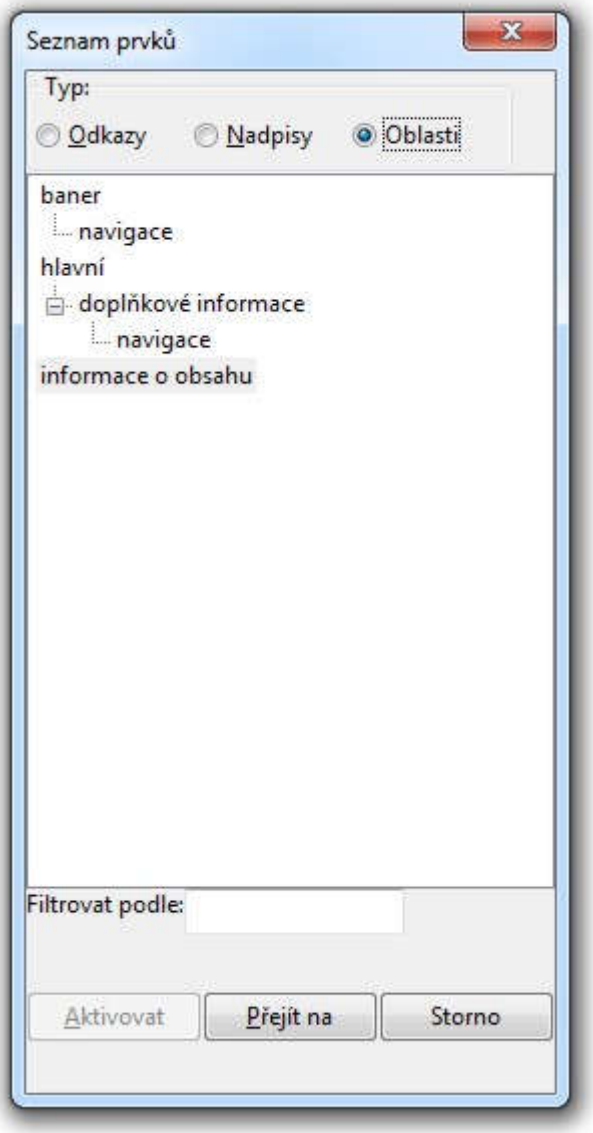
Z tabulky na první pohled plyne, který z prohlížečů dosahuje největší podpory. Tím je Chrome, který podporuje všechny testované elementy vyjma jednoho inputu. Oproti tomu IE (Internet Explorer) si neumí poradit s více jak polovinou testovaných subjektů. To má za následek i horší podporu z hlediska přístupnosti. Firefox je na tom s podporou elementů přibližně stejně jako Chrome. Z uvedených elementů nepodporuje pouze details s potomkem summary. Nicméně za Chromem silně zaostává v podpoře nových intputů, u kterých nepodporuje více jak polovinu, většina se týká vkládání data a času.

8.1.1 Testování pomocí NVDA

Po otestování podpory u zmíněných prohlížečů byla stránka podrobena dalšímu testování, které se tentokrát soustředilo na přístupnost. K tomu dobře posloužil odečítač obrazovky NVDA, jenž poskytl informace o tom, jak by byli hendikepovaní uživatelé informováni o výskytu a chování jednotlivých elementů. Nejprve přišly na řadu oblasti stránky, které

podle posledních statistik z portálu Webaim využívá k navigaci (vždy když jsou k dispozici) necelá ¼ respondentů a pouze 17% je nevyužívá vůbec. Z těchto výsledků lze usoudit, že hendikepovaným uživatelům záleží na tom, aby jim odečítač obrazovky uměl dané oblasti přinejmenším zobrazit.

Testování probíhalo následovně: Otevřela se stránka jednotlivě ve všech prohlížečích a pomocí NVDA se skrze klávesovou zkratku Insert+F7 vyvolala nabídka se seznamem prvků. Tato nabídka obsahuje výpis všech odkazů, nadpisů a oblastí, které se na stránce vyskytují. Na obrázku níže je vidět, jak tato nabídka vypadá a jaké oblasti NVDA zobrazila.



Obrázek 3 - NVDA zobrazení oblastí pomocí nativních elementů (autor)

Toto zobrazení platí pouze u prohlížečů Chrome a Firefox, v případě IE zůstává tato sekce prázdná. Důvodem je implicitní ARIA sémantika (viz. 7.1 Implicitní ARIA sémantika), kterou tyto elementy vlastní. O její zastoupení se starají prohlížeče. IE tuto funkci neplní a neposkytuje tak z hlediska sémantiky o těchto elementech žádné informace, které by mohla asisitvní technologie (v našem případě NVDA) zpracovat. Z těchto důvodů je u elementů vyznačujících oblasti stránky vhodné sémantiku vyjádřit ručně pomocí ARIA atributů. Přestože pak vzniká redundantní kód, je důležité, aby hendikepovaný uživatel měl k této funkci přístup a mohl ji využít v jakémkoliv prohlížeči.

8.1.2 Ostatní elementy a formulářové prvky

U ostatních elementů a formulářových prvků bylo zjišťováno, jak je odečítač obrazovky přečte. NVDA umožňuje zobrazit okno, ve kterém zobrazí, co právě čte. Bylo tak snazší zaznamenat přesnou frázi. Jazyk byl nastavený na český.

Figure a figcaption

Testovaný element Figure obsahoval jeden obrázek a popisek vyplněný ve figcaption. U Firefoxu a IE byl výsledek stejný, jakmile NVDA narazila na tento element, přečetla následující:

Grafika - HTML5 Logo - Oficiální logo HTML5

Grafika demonstruje element img, „HTML5 Logo“ atribut alt a „Oficiální logo HTML5“ text uvnitř elementu figcaption. U Chromu odečítač přečetl pouze text uvnitř figcaption.

Progress a meter

Progress, který slouží k vyznačení hodnoty průběhu, fungoval ve všech prohlížečích stejně. NVDA při jeho nálezu oznámila: „Indikátor průběhu - 50%“. Indikátor průběhu představuje prvek, 50% jeho aktuální hodnotu.

U elementu meter, jenž podporuje Firefox a Chrome, byl výstup stejný jako u elementu progress.

Input - range

Tento input vytvoří slider (posuvník). V jeho případě vypadal výstup takto: „Posuvník - 50“. Při posouvání posuvníku by NVDA měla oznámit aktuální hodnotu. U IE a Firefoxu nebyl s oznámením aktuální hodnoty žádný problém, v případě Chromu NVDA při pohybu posuvníkem neoznámila vůbec nic.

Inputy tel, url, email, search a element datalist

V případě těchto inputů (pokud je prohlížeč podporoval) NVDA oznámila, že se jedná o editační pole. U datalistu, který se vkládá pomocí inputu list, pak oznámila položku jako rozbalovací pole.

Input – number

Input typu number funguje podobně jako typ range, umožňuje uživateli vložit číslo pomocí postraních tlačítek. U obou prohlížečů Chrome a Firefox oznámila NVDA to samé, tedy že se jedná o „okrouhlé“ tlačítko s hodnotou 50. IE tento prvek nepodporuje.

Input – color

Jde o tlačítko, které slouží k vybrání barvy přes dialogové okno. U Firefoxu odečítač NVDA oznámil prvek jako tlačítko, v případě Chromu přečetl aktuálně vybranou barvu, která je ve výchozím nastavení černá. Výstup vypadal takto: 0% red 0% green 0% blue.

Output

U elementu output se NVDA ozvala pouze v případě Chromu, a pouze tehdy když se jeho hodnota změnila.

Časové typy inputů

Časové typy inputů podporuje pouze Chrome. Jakmile uživatel v případě typu `date` (ostatní fungují podobně) narazí na tuto položku, NVDA uživateli přečte, že jde o datum. Uživatel může pomocí tabulátoru postupně přecházet mezi čtyřmi stavy. U prvních tří jde o okrouhlé tlačítko, které postupně nastavuje den-měsíc-rok (hodnotu lze vepsat ručně nebo ji zvýšit/snížit šipkami), následuje tlačítko pro vyvolání podnabídky, které uživateli vyvolá kalendář, z něhož může vybrat celé datum.

Details a summary

Tyto elementy slouží pro zobrazení, potažmo skrytí obsahu a fungují pouze v Chromu. Z hlediska přístupnosti tyto elementy moc nefungují, jakmile uživatel získá fokus na element summary, které se chová jako tlačítko, NVDA oznámí: „Tlačítko - sbalené“. Jakoukoli další interakci, kterou uživatel provede, NVDA nezaznamená.

Time

U elementu time přečetla NVDA pouze text obsažený uvnitř.

Video

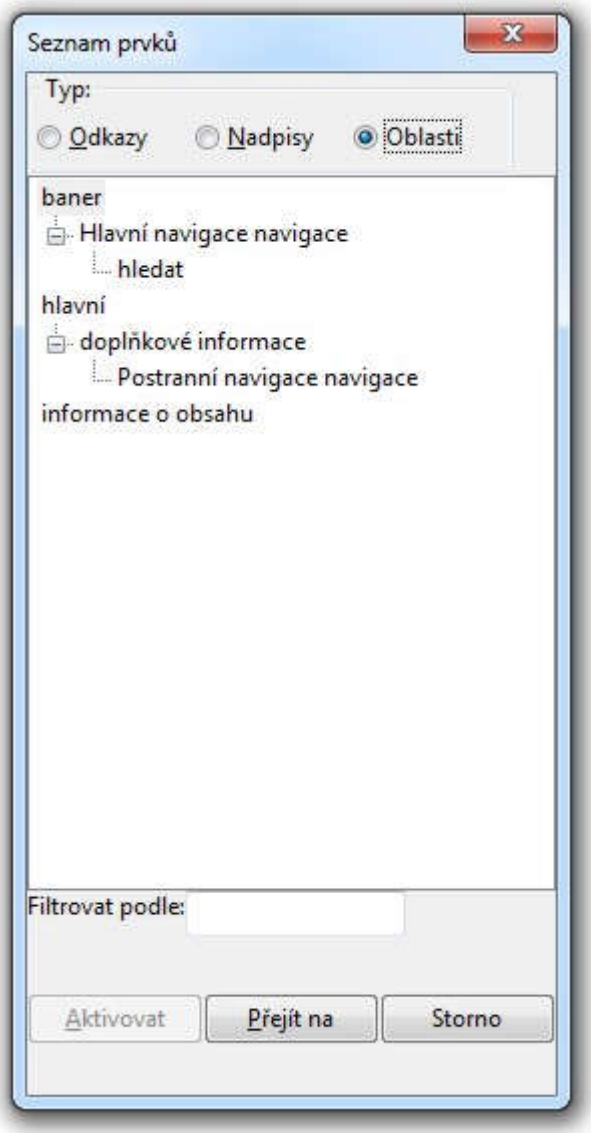
U nativního elementu pro vložení videa, dopadlo testování nejhůře pro Firefox. U Firefoxu se není možné pohybovat po ovládacím panelu videa a přestože skrze klávesnici element získá fokus, pohyb mezi tlačítky není umožněn. U dvou zbylých prohlížečů získají všechna kontrolní tlačítka fokus a uživateli je tak plně poskytnuto jejich ovládání a kontrola videa.

8.2 Využití ARIA

ARIA umožňuje přidat sémantiku tam, kde samotné HTML nestačí, proto se druhé testování zaměřilo na její atributy. Druhá stránka obsahuje pár statických příkladů, které demonstrují, jak ARIA funguje.

8.2.1 Oblasti stránky

První část je opět zaměřena na oblasti stránky. Tentokrát k jejich vyznačení posloužil nesémantický element div obohacený o ARIA atribut s příslušnou rolí, která danou oblast označuje. Oblasti byly rozděleny stejně, jako v případě první stránky. Výpis jednotlivých oblastí v NVDA vypadal následovně:



Obrázek 4 - NVDA zobrazení oblastí skzre ARIA landmarky (autor)

Oproti předchozí variantě přibyla oblast „hledat“, která zastupuje vyhledávací panel. Uživateli je tak umožněno rychle se do této sekce přesunout. Oblast je vyznačena pomocí role „search“, která v HTML nemá podobu nativního elementu.

Vyjma další oblasti, kterou lze pomocí ARIA vyznačit, obrázek demonstruje, k čemu jsou vlastnosti `aria-label` a `aria-labelledby` a kde se uplatní jejich použití. Oba atributy slouží k pojmenování prvků, jejich přesnou funkcionalitu popisuje kapitola – 6.3 Popis elementů a vlastních komponent. V tomto případě bylo záměrem rozlišit dvě navigační

sekce, které jsou v dokumentu obsaženy, uživatel tak získá lepší představu o tom, kde se právě nachází.

8.2.2 Využití ARIA pro definování komponent

Přestože v HTML5 oproti předchůdci přibyla řada elementů, které plní určitou funkci, nepokrývá zdaleka všechny komponenty a ovládací prvky, které se na většině webových stránek vyskytují. I v případě nativních elementů nebývá jejich použití vždy možné, a to z důvodu chabé implementace v prohlížečích nebo z nedostatečné funkce, kterou daný element poskytuje. Z těchto důvodů bývají často využívány elementy `div` a `span`, které ovládací prvek zastupují. Jelikož oba tyto elementy nemají sémantický význam, je ovládací prvek, který zastupují, pro asistivní technologii nerozeznatelný. Elementům lze přidat sémantiku pomocí ARIA, která pokrývá mnohem širší skupinu ovládacích prvků.

Pro demonstraci, jak lze za pomocí ARIA vytvořit sémanticky viditelný ovládací prvek, následuje několik příkladů.

Progressbar a slider

Přestože obě tyto komponenty mají nativní zastoupení v HTML, progressbar skrze element `progress` a slider pomocí `<input type="range">`, existují situace, které jejich použití neumožňují, např. kvůli stylizaci nebo z důvodu, že implementace vlastní komponenty je v dané situaci vhodnější. Při definování vlastního progressbaru tak, aby jej asistivní technologie dokázala rozpoznat, je nutné definovat řadu atributů.

```
<div role="progressbar" aria-valuenow="50" aria-valuemin="0" aria-
valuemax="100"></div>
```

- Atribut `role` změní význam elementu `div` na progressbar
- `Aria-valuenow` poskytuje informace o aktuální hodnotě
- Atributy `aria-valuemin` a `aria-valuemax` informují o minimální a maximální hodnotě, které může progressbar dosáhnout.

Element `div` přejal sémantiku, kterou mu ARIA poskytla a skrze NVDA byl vyjádřen jako „indikátor průběhu“ s hodnotou „50“.

Menu nabídka

Pomocí ARIA lze sémanticky popsat víceúrovňovou nabídku, jako je např. rozbalovací menu. Slouží k tomu následující role: `menubar`, `menu` a `menuitem`. Zatímco `menubar` definuje horizontálně orientovanou nabídku, tak role `menu` definuje vertikální. `Menuitem` označuje jednotlivé položky, které nabídka obsahuje. Důležitým atributem při vytváření sémantické nabídky je také atribut `aria-haspopup`. Tento atribut se používá pro vytváření podnabídek.

NVDA u následujícího příkladu ohlásí menu jako „nabídka“ a jednotlivé položky jako „položka nabídky“.

```
<div role="menu">
  <div role="menuitem">Nový</div>
  <div role="menuitem">Otevřit</div>
  <div role="menuitem">Uložit</div>
</div>
```

Seznam

V HTML slouží pro vytvoření seznamu elementy `<ul>` a `<ol>`, jednotlivé položky seznamu tvoří element `<li>`. ARIA umožňuje docílit podobného efektu s rolí `list`, jenž definuje seznam, a rolí `listitem`, která definuje položku seznamu. Pro srovnání vytvořený seznam pomocí nativního HTML a skrze ARIA list:

HTML zápis s použitým element `<ul>`:

```
<ul>
  <li>1</li>
  <li>2</li>
  <li>3</li>
</ul>
```

Využití ARIA role `list` u elementu `div`:

```
<div role="list">
  <div role="listitem">1</div>
  <div role="listitem">2</div>
  <div role="listitem">3</div>
</div>
```

NVDA u obou zápisů oznámí, že se jedná o seznam s třemi položkami.

Tooltip

Tooltip je vyskakovací bublina, která uživateli zobrazí dodatečné informace o prvku. Obvykle se tento popisek zobrazí po najetí myši na daný prvek nebo v případě, že prvek získá fokus. Skrze ARIA lze sémanticky vyjádřit tooltip pomocí role `tooltip` a propojit ho s elementem přes atribut `aria-describedby`. Tooltip lze využít např. u textového pole, které po získání fokusu zobrazí krátký popisek.

```
<div>
  <label for="username">Jméno</label>
  <input id="username" aria-describedby="tip" type="text">
  <span role="tooltip" id="tip" class="tooltip">Vložte své jméno.</span>
</div>
```

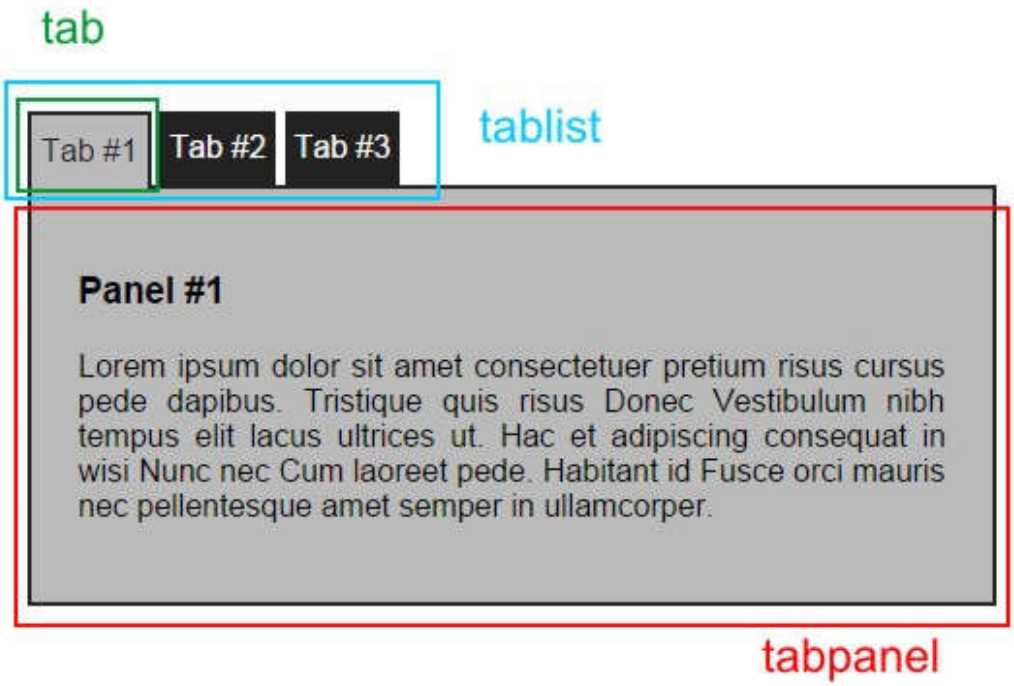
Jakmile je textové pole vybrané NVDA ohlásí název pole „Jméno“ a popisek „Vložte své jméno“.

8.3 Tvorba vlastní komponenty

Předmětem posledního testování je vytvoření funkčního ovládacího prvku (Tab panel), jehož účel spočívá v přepínání karet. V rámci samotného HTML (bez CSS a JavaScriptu) neexistují elementy, které by tento funkční prvek vytvořily a sémanticky zvýraznily. ARIA umí tento prvek popsat velmi dobře za pomoci následujících rolí:

- `tablist` – popisuje kontejner, který obsahuje jednotlivé záložky
- `tab` – slouží k vyznačení záložky
- `tabpanel` – označuje panel, který se zobrazí po vybrání záložky

Následující obrázek znázorňuje využití výše zmíněných rolí.



Obrázek 5 - Tab panel (autor)

Uvedené role umožňují jednotlivé části komponenty rozeznat. Jelikož se obsah komponenty mění podle zvolené záložky, je nutné uživatele určitým způsobem informovat, jaká karta je právě aktivní. K tomu slouží atribut `aria-selected`, který s nastavenou hodnotou na "true" asisitvní technologii oznámí, že zvolená položka je právě aktivní.

Podmínkou přístupnosti je, že veškerý „fokusovatelný“ obsah musí být přístupný z klávesnice. Proto je nutné zařadit Tab panel do „Tab-order“ (pořadí aktivace). K tomu slouží atribut `tabindex`. U aktivní záložky je `tabindex` nastavený na hodnotu "0", tato vlastnost přidá prvek do pořadí aktivace, jinými slovy prvek získá fokus. U nevybraných záložek je `tabindex` roven hodnotě "-1", která prvkům naopak znemožní získat fokus. V praxi to vypadá následovně: Uživatel skrze tabulátor získá fokus na vybranou záložku a dalším kliknutím na klávesu Tab tablist opustí. K přepínání mezi kartami slouží klávesy šipek.

Další atribut, který tato komponenta obsahuje, je `aria-controls`. Tato vlastnost propojí ovládací prvek (záložky) s ovládaným prvkem (panel). K propojení slouží identifikátor. Kompletní zápis v HTML vypadá následovně:

```
<div class="tablist">
  <ul role="tablist">
    <li role="presentation"><a role="tab" href="#panel1" aria-
controls="panel1" tabindex="0" aria-selected="true">Tab #1</a></li>
    <li role="presentation"><a role="tab" href="#panel2" aria-
controls="panel2" tabindex="-1">Tab #2</a></li>
    <li role="presentation"><a role="tab" href="#panel3" aria-
controls="panel3" tabindex="-1">Tab #3</a></li>
  </ul>
  <div id="panel1" role="tabpanel">
    <h3 tabindex="0">Panel #1</h3>
    <p>…</p>
  </div>
  <div id="panel2" role="tabpanel">
    <h3 tabindex="0">Panel #2</h3>
    <p>…</p>
  </div>
  <div id="panel3" role="tabpanel">
    <h3 tabindex="0">Panel #3</h3>
    <p>…</p>
  </div>
</div>
```

V kódu se navíc u položek `<li>` nachází atribut `role="presentation"`, jehož účel je potlačit sémantiku elementu. Sémantika, kterou v tomto případě elementy `<li>` poskytují, není žádoucí a mohla by mít negativní vliv na výstup asistivní technologie.

Funkční skript komponenty je součástí přílohy – Příloha 2.

8.3.1 Chování NVDA a její výstup

Jakmile NVDA narazí na tablist, oznámí jej jako „seznam záložek“. U pohybu mezi záložkami se výstup z odečítače trochu liší, a to podle toho, jaký z prohlížečů je zrovna používán. V případě prohlížečů Chrome a IE je aktivní záložka oznámena jako „Tab #1 záložka vybráno“. U Firefoxu navíc NVDA oznámí i počet záložek a pořadí právě vybrané záložky, např. „Tab #1 záložka vybráno 1 z 3“. Pořadí záložek je v tomto případě závislé na nastaveném atributu `role="presentation"` u elementů `<li>`. Jakmile by tento atribut nebyl nastaven, NVDA ohlásí u všech položek pořadí „1 z 1“. Nastal by tak zmatek v počtu záložek, což by mohlo mít negativní přínos pro hendikepovaného uživatele využívající NVDA.

9 Závěr

Tématem práce bylo poukázat na výhody a nevýhody, které přináší nová sémantika HTML5 a kdy je zapotřebí využít technik WAI-ARIA.

V teoretické části bylo cílem tuto problematiku popsat a jednotlivé pojmy vysvětlit. V úvodu práce byl popsán vývoj jazyka HTML5 a jeho hlavní přínos, následoval popis sémantických elementů a formulářových prvků. Dále byly vysvětleny pojmy přístupnost a asitivní technologie, jejichž pochopení je nezbytné pro práci s nástrojem WAI-ARIA. Následně došlo k definování samotného nástroje, všech jeho vlastností a základních pravidel použití. Na konci teoretické části byl definován vztah a vzájemné propojení jazyka HTML s nástrojem WAI-ARIA.

Cílem praktické části byla analýza sémantiky vytvořené webové stránky, pro tyto účely byla vytvořena celkem třikrát. Nejprve byla zhotovena za pomoci nových HTML5 elementů, poté s využitím technik WAI-ARIA a nakonec kombinací obou technologií na funkční komponentě.

Začátek praktické části se zaměřil na podporu použitých HTML5 elementů u webových prohlížečů, následně byly tyto elementy v prohlížečích testovány z hlediska přístupnosti. Další kapitola obsahovala využití WAI-ARIA, testovaly se oblasti stránky a několik příkladů statických komponent. Poslední část obsahovala propojení HTML a WAI-ARIA při tvorbě vlastní komponenty. Zvolenou komponentou byl Tab panel, na kterém bylo demonstrováno praktické použití WAI-ARIA atributů, pro vyjádření sémantiky a zpřístupnění ovládacího prvku asistivním technologiím. Přístupnost a použitelnost testoval odečítač obrazovky NVDA.

V praktické části bylo analýzou u vytvořených webových stránek zjištěno, že ačkoliv HTML5 poskytuje několik nových možností, jak vytvořit a sémanticky popsat různé komponenty, tak nepokrývá zdaleka všechny, které se běžně na webových stránkách vyskytují. Zároveň použití nových elementů není vždy vhodné. Hlavním důvodem je podpora v prohlížečích, která je u některých z nich stále nedostatečná, a dále pak omezená funkce, kterou daný prvek disponuje. Z těchto důvodů vývojářům obvykle nezbývá jiná možnost, než definovat vlastní prvky, které danou komponentu zastoupí, a jejich význam se pokusit vyjádřit pomocí atributů WAI-ARIA.

Jelikož jde v tomto odvětví vývoj neustále kupředu, dá se očekávat, že postupem času přibudou elementy, které pokryjí další komponenty a ovládací prvky. Prozatím se vývojáři musí omezit na to, co jim HTML5 nabízí právě teď a případné problémy s přístupností řešit skrze nástroj WAI-ARIA.

Literatura

1. STEVENS, Luke a RJ OWEN. The truth about HTML5. New York: Apress, 2013. ISBN 978-1-4302-6415-6.
2. LAWSON, Bruce a Remy SHARP. Introducing HTML 5. 2nd ed. Berkeley, CA: New Riders, 2012, xvi, 295 p. Voices that matter. ISBN 03-217-8442-1.
3. W3C. HTML5. W3C. [online]. 28.10.2014 [cit. 2015-09-15]. Dostupné z: http://www.w3.org/TR/html5/
4. WHATWG . HTML Living Standard. WHATWG. [online]. © 2015 [cit. 2015-10-26]. Dostupné z: https://html.spec.whatwg.org/#is-this-html5?
5. LAWSON, Bruce. On HTML5 vs Living Standard, W3C vs WHATWG. Bruce Lawson. [online]. 28.10.2014 [cit. 2015-10-26]. Dostupné z: http://www.brucelawson.co.uk/2014/on-html5-vs-living-standard-w3c-vs-whatwg/
6. MOZILLA. HTML5. Mozilla Developer Network. [online]. © 2005-2015 [cit. 2015-10-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5
7. CONNOR, Joshue O. Pro HTML5 accessibility: building an inclusive web. New York: Distributed to the book trade worldwide by Springer Science Business Media, 2012, xix, 365 p. ISBN 978-1430241942.
8. WEBAIM. Screen Reader User Survey #6 Results. Webaim. [online]. 28.8.2015 [cit. 2016-01-03]. Dostupné z: http://webaim.org/projects/screenreadersurvey6/
9. PILGRIM, Mark. Ponořme se do HTML5. Praha: CZ.NIC, z.s.p.o., 2015, 278 stran. CZ.NIC. ISBN 978-80-905802-6-8.
10. LEADBETTER, Tom. Semantic navigation with the nav element. HTML5 Doctor. [online]. 15.7.2009 [cit. 2015-11-02]. Dostupné z: http://html5doctor.com/nav-element/
11. CLARK, Richard. The main element. HTML5 Doctor. [online]. 26.6.2013 [cit. 2015-11-03]. Dostupné z: http://html5doctor.com/the-main-element/
12. CLARK, Richard. The figure & figcaption elements. HTML5 Doctor. [online]. 13.4.2010 [cit. 2015-11-03]. Dostupné z: http://html5doctor.com/the-figure-figcaption-elements/
13. LEADBETTER, Tom. The details and summary elements . HTML5 Doctor. [online]. 9.8.2011 [cit. 2015-11-03]. Dostupné z: http://html5doctor.com/the-details-and-summary-elements/

14. MOZILLA. Time. Mozilla Developer Network. [online]. © 2005-2015 [cit. 2015-11-06]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/time
15. CLARK, Richard. HTML5 forms input types. HTML5 Doctor. [online]. 28.2.2013 [cit. 2015-12-12]. Dostupné z: http://html5doctor.com/html5-forms-input-types/
16. DOBRÝ WEB. O přístupnosti. Přístupnost.cz. [online]. © 2010 [cit. 2015-11-18]. Dostupné z: http://www.pristupnost.cz/o-pristupnosti/
17. HENRY, Shawn. Introduction to Web Accessibility. W3C – Web Accessibility Initiative. [online]. © 1994-2012 [cit. 2015-11-19]. Dostupné z: http://www.w3.org/WAI/intro/accessibility.php
18. PAVLÍČEK, Radek. Přístupný web a jak se vyvarovat chyb. Ministerstvo vnitra České republiky. [online]. 2009 [cit. 2015-11-19]. Dostupné z: http://www.mvcr.cz/clanek/pristupny-web-a-jak-se-vyvarovat-chyb.aspx
19. HENRY, Shawn. Accessibility. W3C. [online]. © 2015 [cit. 2015-11-20]. Dostupné z: http://www.w3.org/standards/webdesign/accessibility
20. PAVLÍČEK, Radek. Termín asistivní technologie pohledem Radka Seiferta. Poslepu. [online]. 22.9.2014 [cit. 2016-1-3]. Dostupné z: http://poslepu.cz/termin-asistivni-technologie-pohledem-radka-seiferta/
21. PAVLÍČEK, Radek. WAI ARIA 1.0 byla vydána jako doporučení W3C. Poslepu. [online]. 15.4.2014 [cit. 2015-10-30]. Dostupné z: http://poslepu.cz/wai-aria-1-0-byla-vydana-jako-doporuceni-w3c/
22. WEBAIM. Accessibility of Rich Internet Applications. WebAIM. [online]. 28.8.2013 [cit. 2015-11-10]. Dostupné z: http://webaim.org/techniques/aria/
23. MAX, Stephan. An Introduction to WAI-ARIA. Sitepoint. [online]. 21.7.2014 [cit. 2015-11-12]. Dostupné z: http://www.sitepoint.com/introduction-wai-aria/
24. W3C. Notes on Using ARIA in HTML. World Wide Web Consortium (W3C) [online]. 21.5.2015 [cit. 2015-12-21]. Dostupné z: http://www.w3.org/TR/aria-in-html/
25. FAULKNER, Steve. Using WAI-ARIA Landmarks – 2013. Paciellogroup. [online]. 12.2.2013 [cit. 2015-12-21]. Dostupné z: https://www.paciellogroup.com/blog/2013/02/using-wai-aria-landmarks-2013/
26. PICKERING, Heydon . UX accessibility with aria-label. Dev.Opera. [online]. 9.4.2015 [cit. 2015-12-20]. Dostupné z: https://dev.opera.com/articles/ux-accessibility-aria-label/

27. JAHODA, Bohumil. ARIA atributy. Je čas. [online]. 10.5.2015 [cit. 2015-12-20]. Dostupné z: http://jecas.cz/aria
28. FAULKNER, Steve. On HTML belts and ARIA braces (The Default Implicit ARIA semantics they didn't want you to know about). HTML5 Doctor. [online]. 14.4.2015 [cit. 2015-12-27]. Dostupné z: http://html5doctor.com/on-html-belts-and-aria-braces/
29. ANSARI, Rafay Saeed. Avoiding Redundancy with WAI-ARIA in HTML Pages. Sitepoint. [online]. 15.8.2015 [cit. 2015-12-27]. Dostupné z: http://www.sitepoint.com/avoiding-redundancy-wai-aria-html-pages/
30. FAULKNER, Steve. ARIA in HTML – there goes the neighborhood. Paciellogroup. [online]. 1.10.2014 [cit. 2015-12-28]. Dostupné z: https://www.paciellogroup.com/blog/2014/10/aria-in-html-there-goes-the-neighborhood/

Seznam obrázků

Seznam tabulek

Seznam příloh

Příloha 1 – Tabulka podporovaných elementů u webových prohlížečů

HTML5 elementy	Chrome	Firefox	Internet Explorer
Article	ok	ok	ok
Section	ok	ok	ok
Aside	ok	ok	ok
Header	ok	ok	ok
Footer	ok	ok	ok
Nav	ok	ok	ok
Main	ok	ok	-
Figure	ok	ok	ok
Figcaption	ok	ok	ok
Details	ok	-	-
Summary	ok	-	-
Time	ok	ok	-
Output	ok	ok	-
Datalist	ok	ok	ok
Progress	ok	ok	ok
Meter	ok	ok	-
Inputy			
Tel	-	-	-
Url	ok	ok	ok
Email	ok	ok	ok
Search	ok	-	-
Date	ok	-	-
Month	ok	-	-
Week	ok	-	-
Time	ok	-	-
Datetime-local	ok	-	-
Number	ok	ok	-
Range	ok	ok	ok
Color	ok	ok	-

Příloha 2 – Skript k Tab panelu

```
//Ovládání pomocí klávesnice
$('.tablist li a').on('keydown', function(e) {
  var $original = $(this);
  var $prev = $(this).parents('li').prev().children('.tablist a');
  var $next = $(this).parents('li').next().children('.tablist a');
  var $target;

  // Přepínání
  switch (e.keyCode) {
    case 37:
      $target = $prev;
      break;
    case 39:
      $target = $next;
      break;
    default:
      $target = false
      break;
  }
  if ($target.length) {
      $original.attr({
        'tabindex' : '-1',
        'aria-selected' : null
      });
      $target.attr({
        'tabindex' : '0',
        'aria-selected' : true
      }).focus();
  }
  //Přepnutí panelů (skryje a odkryje panel)
  $('.tablist div').hide();
  $('#' + $(document.activeElement).attr('href').substring(1)).show();
});

//Přepínaní panelů (ovládání pomocí myši)
$('.tablist a').on('click', function(e) {
  e.preventDefault();
  //Nastavení atributů tabindex a aria-selected při kliknutí na položku
  $('.tablist a').attr({
    'tabindex' : '-1',
    'aria-selected' : null
  });
  $(this).attr({
    'tabindex' : '0',
    'aria-selected' : true
  });
  //Přepnutí panelů (skryje a odkryje panel)
  $('.tablist div').hide();
  $('#' + $(this).attr('href').substring(1)).show();
});
```

Příloha 3 – Soubory webových stránek

Soubory webových stránek jsou uloženy na této adrese:

<https://drive.google.com/open?id=0Byg7MalLIe-KUjRlcVFvQVJLbjA>