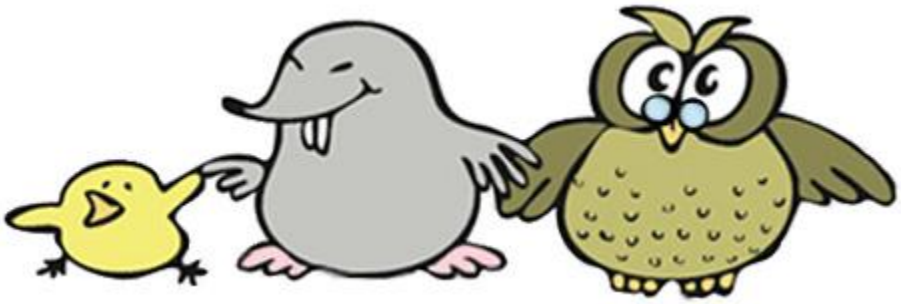
Bokprosjekt

En guide til WAI-ARIA

Gruppe 28

5/26/2012



HØGSKOLEN I OSLO OG AKERSHUS
Studieprogram: Informasjonsteknologi
Postadresse: Postboks 4 St. Olavs plass, 0130 Oslo
Besøksadresse: Holbergs plass, Oslo

Telefon: 22 45 32 00
Telefaks: 22 45 32 05

PROSJEKT NR.
28

TILGJENGELIGHET
Åpen

HOVEDPROSJEKT

HOVEDPROSJEKTETS TITTEL	DATO
EN GUIDE TIL WAI-ARIA	26.05.2012
	ANTALL SIDER / BILAG: 13
PROSJEKTDELTAKERE: Kent-Remi Fredriksen, Anders Knaplund Pettersen, Evy Litovchenco, Ivar E. Røyse	INTERN VEILEDER: Weiqin Chen

OPPDRAGSGIVER	KONTAKTPERSON
MediaLT	Morten Tollefsen

SAMMENDRAG

Dette er prosjektdokumentasjon for Gruppe 28 ved Høyskolen i Oslo våren 2012.
Rapporten består av Prosessdokumentasjon, Produktdokumentasjon, Testdokumentasjon, Kildehenvisninger, Vedlegg og Brukerveiledning.

Resultatet er en web-basert guide på funksjonalitet til W3C sin universelle utformingsstandard WAI-ARIA laget i henhold til en bok om universell utforming som skrives av Morten Tollefsen fra MediaLT. Nettsiden finner du på www.aria.moo.no (www.stud.hio.no/~s161868/).

Nettsiden er hovedsakelig utviklet i HTML5, CSS3, PHP, JavaScript og jQuery.

3 STIKKORD
Universell Utforming
WAI-ARIA
Rike applikasjoner

Innholdsliste

Prosessdokumentasjon

Forord

Dette er en bacheloroppgave utført av fire studenter ved Høgskolen i Oslo og Akershus våren 2012 i oppdrag fra MediaLT. Prosjektet gir 20 studiepoeng til sammen. Tre av studentene tar Anvendt Datateknologi, og en tar Informasjonsteknologi.

Denne rapporten er en dokumentasjon fra Gruppe 28 sitt hovedprosjekt ved Høyskolen i Oslo og Akershus.

Dette dokumentet er inndelt på følgende måte;

- **Innledning,** en introduksjon av prosjektet.
- **Planlegging og metoder,** planleggelsen av prosjektet og metoder og hjelpemidler brukt for å fullføre prosjektet.
- **Om utviklingsprosessen,** gjennomføringen av produktutviklingen fra start til slutt.
- **Resultatet,** resultat og konklusjon av prosjektet.
- **Avslutning,** en oppsummering av prosjektet og drøfting av resultatet.

Dette dokumentet er ment for sensor, oppdragsgiver og andre som er interesserte i universell utforming på web. Det er en fordel om leseren har noe forståelse av HTML og konstruksjon av nettsider og universell utforming.

Gruppe 28 vil gjerne først og fremst takke MediaLT og Morten Tollefsen for å bistå med dette prosjektet og for all hjelp de har bidratt med. Så vil vi gjerne takke vår støttende veileder Weiqin Chen ved Høgskolen i Oslo og Akerhus som har vært en hjelpende hånd rundt hele denne prosessen. Til slutt vil vi gjerne takke vårt testpanel for å ta seg tid til å brukerteste nettstedet.

Innholdsfortegnelse

Innholdsfortegnelse

1. Innledning

Alle på gruppen ønsket å jobbe med noe innenfor brukervennlighet, og vi tok selv kontakt med MediaLT som hadde annonsert at de ønsket å samarbeide med studenter. De hadde flere prosjekter vi kunne bidra til, men det var størst enighet om bokprosjektet. Oppdraget ga litt frie tøyler og det så ut som et interessant og morsomt prosjekt. I tillegg er universell utforming noe vi hadde friskt i minne da tre av oss akkurat hadde fullført et semesterfag i samme tema. Vi hadde likevel begrenset kjennskap til blant annet skjermlesere og lignende verktøy fra tidligere.

1.1 Gruppen

Fullt navn	**Studentnummer**	**Kull**	**E-post**
Ivar E. Røise	s148181	3IA	s148181@stud.hioa.no
Kent-Remi Fredriksen	s161890	3DA	s161890@stud.hioa.no
Anders Knaplund Pettersen	s161876	3DB	s161876@stud.hioa.no
Evy Litovchenco	s161868	3DA	s161868@stud.hioa.no

1.2 Oppdragsgiver

MediaLT

Adresse: Jerikoveien 22, 1067 Oslo
Telefon: (+47) 21 53 80 1
E-post: info@medialt.no
Vev: http://www.medialt.no/

MediaLT driver med opplæring, tilrettelegging og utvikling for funksjonhemmede, og er en ledende kompetansebedrift innen IKT og universell utforming.

Kontaktpersoner:

Morten Tollefsen
Forskningsleder
E-post: morten@medialt.no
Telefon: (+47) 90 89 93 05

Magne Lunde
Daglig leder
E-post: magne@medialt.no
Telefon: (+47) 91 79 00 78

1.3 Dagens situasjon

"Universell utforming er utforming av produkter og omgivelser på en slik måte at de kan brukes av alle mennesker, i så stor utstrekning som mulig, uten behov for tilpassing og en spesiell utforming."[1] (ref)

I dag er nettsider i Norge i liten grad universelt utformet. Utviklingen mot stadig mer interaktive og kompliserte nettsider fører til redusert tilgjengelighet for brukere som av forskjellige grunner er forhindret fra å bruke nettsider på den vanligste måten; visuelt og med mus. Spørsmålet om universell utforming av IKT-systemer, herunder nettsider, ble tatt opp i forbindelse med arbeidet som ledet fram til Diskriminerings- og tilgjengelighetsloven av 2008 (Dtl.).

I Dtl. § 11 heter det: *"Nye IKT-løsninger som [... er ...] stillet til rådighet for allmennheten, skal være universelt utformet fra og med 1. juli 2011, men likevel tidligst tolv måneder etter at det foreligger standarder eller retningslinjer for innholdet i plikten. For eksisterende IKT-løsninger gjelder plikten fra 1. januar 2021."*

De omtalte standardene og retningslinjene er ennå ikke fastlagt, det er dermed ikke klart hva innholdet blir. Slike retningslinjer kan imidlertid være av to slag. De kan fastslå hvilken *funksjon* som skal oppnås, eller de kan spesifisere hvilke *teknikker* som skal brukes. Retningslinjene kan selvfølgelig også inneholde krav av begge typer. Gitt den raske utviklingen innen feltet, vil retningslinjer som kun inneholder teknikker, ganske snart bli utdaterte. Videre er funksjonen det viktigste for dem som har mest nytte og behov for universell utforming. Vi har dermed god grunn til å tro at de kommende retningslinjene først og fremst vil fokusere på funksjon.

Dette prosjektet dreier seg om et avgrenset område av IKT-feltet, nemlig det som dreier seg om nettsider. På dette området finnes det allerede internasjonalt etablerte retningslinjer

[1] http://www.bufetat.no/bufdir/deltasenteret/Universell-utforming/

(WCAG[2]) som utgjør en de facto standard for universell utforming. WCAG er utgitt av The World Wide Web Consortium (W3C[3]).

I Norge har Direktoratet for forvaltning og IKT gjennom 10 år utført årlige kvalitetskontroller på offentlige nettsteder, der en vesentlig del av kvalitetskriteriene[4] er knyttet til tilgjengelighet. Bufetat v/Deltasenteret har utgitt en veileder for tilgjengelige nettsteder[5] i tre deler. Som et tillegg til den veilederen har Helsedirektoratet utgitt en utformingsveileder for kognitiv tilgjengelighet av nettsider og nettsteder.[6] Felles for de foreliggende norske retningslinjene er at de i stor grad sammenfaller med WCAG.
WCAG er orientert mot funksjon, som beskrevet ovenfor. Samtidig inneholder dokumentet eksempler på teknikker som kan brukes for å tilfredsstille kravene som oppstilles.
I mangel av fastsatte forskrifter til Dtl., og etter å ha konstatert at det er lite, om noen, konflikter mellom WCAG og de norske retningslinjene som allerede finnes, bestemte vi oss for å forholde oss kun til WCAG i dette prosjektet. Dette valget begrenser dokumentmengden vi må bry oss om, uten at det går ut over kvaliteten på prosjektet. Det er også forenlig med MediaLTs ønske om å ta utgangspunkt i WCAG når det gjelder nettstedets tilgjengelighet. Et annet vesentlig moment er at det finnes mange hjelpemidler for å verifisere at nettsider følger WCAG, noe som letter utviklingen av nettstedet. At WCAG-dokumentene er godt opplenket mot andre dokumenter og viktige standarder hos W3C, er også en stor fordel.

Med stadig mer (inter)aktive og dynamiske nettsider ble det klart at det trengtes tekniske hjelpemidler for å kunne tilfredstille WCAG. Det medførte at W3C utviklet et rammeverk for å gjøre dynamiske nettsider/-steder universelt tilgjengelige. Dette rammeverket kalles Web Accessibility Initiative - the Accessible Rich Internet Applications Suite. Heretter kun

---

[2] Web Content Accessibility Guidelines. Norsk autorisert oversettelse: http://www.w3.org/Translations/WCAG20-no-20110926/

[3] http://www.w3.org/Consortium/

[4] http://kvalitet.difi.no/kriteriesett/

[5] http://www.bufetat.no/bufdir/deltasenteret/publikasjoner/

[6] http://iktforalle.no/veileder-deler.html

WAI-ARIA eller helt enkelt ARIA. WAI-ARIA er et tillegg til HTML-standardene, som lar kompatible nettlesere og støtteteknologi presentere dynamisk og interaktivt innhold på en meningsfylt måte. Det må her nevnes at uten et slikt omforent rammeverk, vil det være svært vanskelig, kanskje umulig, å produsere plattformuavhengige universelt tilgjengelige nettsider.

I dag er WAI-ARIA veldig lite i bruk i Norge, og kjennskapen til det er generelt lite utbredt. For å være i tråd med kravene fra Dtl. bør denne situasjonen forbedres. Innholdsleverandører vil bli nødt til å bruke WAI-ARIA for å tilfredsstille kravene i Dtl., dersom de fremdeles vil tilby dynamisk innhold. Og det må man anta at de vil. Her vil de dermed ha en vesentlig jobb å gjøre. Bruken av WAI-ARIA er ikke veldig vanskelig eller komplisert, men det er enklere å gjøre det i startfasen, enn å implementere i ettertid.

MediaLT håper på å lage en lærebok om universell utforming av nettsider ved hjelp av WAI-ARIA, og ønsker derfor en ressursside med eksempler som viser løsninger på forskjellige utfordringer knyttet til rike webapplikasjoner på nett. Applikasjonene vil hovedsakelig være utviklet i HTML5 og JavaScript (med jQuery) hvor det legges fokus på tilrettelegging for funksjonshemmede brukere, men vi tar og til høyde for at det skal kunne fungere så godt som mulig i tidligere HTML-versjoner. HTML5 tilbyr dog ny og nyttig funksjonalitet for bedre tilpasning av applikasjoner på nett.

2. Planlegging og metode

2.1 Planlegging

Tidlig i prosjektet utarbeidet vi en risikoplan:

Risiko	**Sannsynlighet**	**Konsekvens**	**Tiltak**
Dårlig planlegging	Moderat	Alvorlig	Må lage kontrakt så snart som mulig. Faste møter.
Interne konflikter	Lav	Alvorlig	Ta opp irritasjonsmomenter jevnlig
Feil forståelse av oppgaven	Moderat	Alvorlig	Viktig å samarbeide tett med oppdragsgiver
Sykdom (kortvarig)	Veldig høy	Akseptabel	Ta tran og huske lue. Er man blitt syk holder man seg hjemme
Sykdom (langvarig)	Moderat	Alvorlig	Finne en erstatter, evt få utsettelse fra oppdragsgiver om det er meget kritisk
Tap av data	Lav	Katastrofal	Lav sannsynlighet for å tape data ettersom alt lagres online hos alle prosjektmedlemmer i reviderte versjoner
Uengasjert oppdragsgiver	Lav	Middels	Oppdragsgiver har mye erfaring med hovedprosjekter, men hvis de ikke stiller opp med hjelp vil vi måtte kontakte veileder for hjelp
Gruppemedlem	Lav	Alvorlig	Diskutere med

slutter			oppdragsgiver/veileder

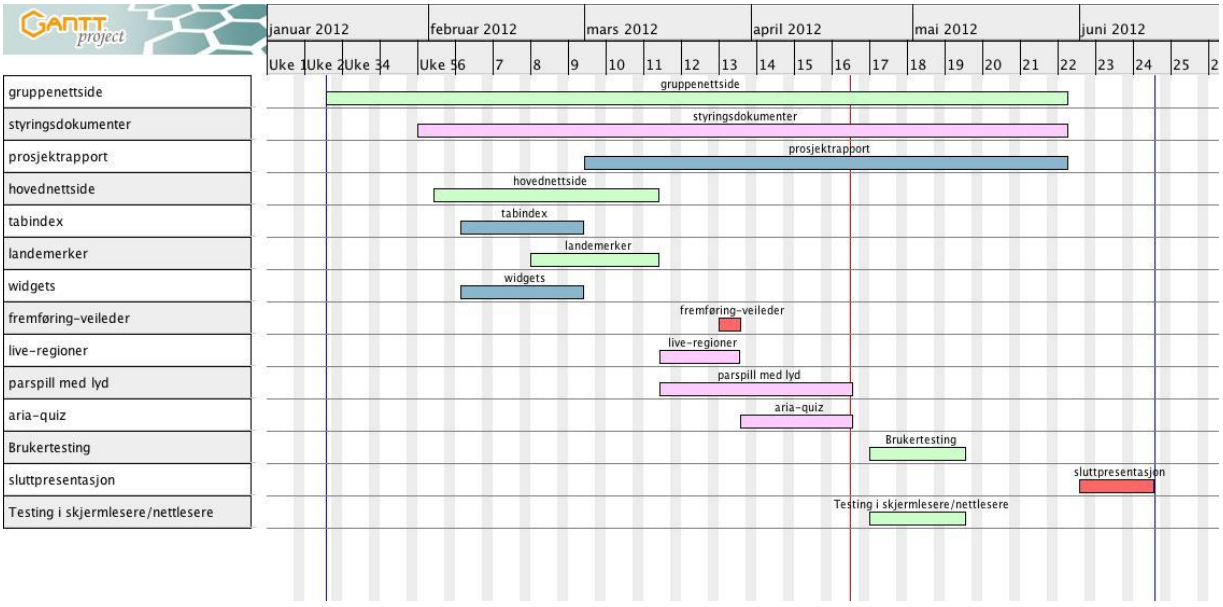
I tillegg utarbeidet vi en kontrakt som ble signert av alle:

Ved å signere denne kontrakten sier du deg enig i:

- *Å bidra under hele prosjektperioden.*
- *Å yte ditt beste under hele prosjektperioden etter dine kunnskaper og muligheter.*
- *Å møte opp til alle planlagte møter, om så ikke er mulig skal det gis beskjed før oppmøte.*
- *Å ta eget initiativ og være selvstendig arbeidsdrivende.*

2.1.1 Tidsplanlegging

Vi opprettet et gantt-diagram i Gantt-project for å overholde tidsfrister for hver større iterasjon. Dette diagrammet måtte endres litt underveis ettersom arbeidsoppgavene endret seg. Diagrammet ble utformet med tanke på at alle demonstrasjonene skulle ta like lang tid, men det viste seg at det var varierende vanskelighetsgrad på mange av dem.



2.1.3 Samarbeid med oppdragsgiver

Hver fredag har vi jobbet fast ute hos MediaLT. Dette ga en fin oppsummering av uken og vi fikk lagt planer for påfølgende uke, og vi fikk også hjelp av oppdragsgiver på områder der vi stod fast. Ellers har vi hatt hyppig kontakt via mail underveis i prosjektet.

2.1.4 Utfordringer

Det var veldig vanskelig å booke grupperom underveis i prosjektet siden alle rom har vært okkuperte på dagtid. Vi har stort sett jobbet på en bråkete datalab eller funnet auditorium som har vært ledige, men her ble vi ofte sendt rundt fra rom til rom og brukte mye tid på å finne andre steder på høyskolen. Fredagene vi tilbrakte på MediaLT var veldig produktive da vi fikk jobbe i ro på et møterom de hadde til rådighet. Ettersom bedriften hadde mye møter ellers i uka var det stort sett fredag vi fikk tilgang til å sitte der.

2.2 Utviklingsmetode

Som utviklingsmetode ble vi enige om å bruke SCRUM, ettersom dette er en en metode som virker utbredt i arbeidslivet og det passet bra i forhold til vårt prosjekt ettersom det arbeides inkrementelt og iterativt, man får sett resultater ganske fort og man har et tett samarbeid med kunden (Schwaber, & Sutherland, 2008).

SCRUM er et utviklingsrammeverk basert på leveranser i små iterasjoner på en fast tidslengde. Disse kalles sprints og kan variere fra en uke til en måned. SCRUM utføres i små grupper, SCRUM-team, med en leder, SCRUM-master. Gruppen har daglig korte møter for å oppdatere status på sprinten. Før hver sprint planlegges funksjoner som skal implementeres fra et sett med krav fra kunden (kravspesifikasjon). Man har et tett samarbeid med kunden og man prioriterer funksjonene kunden verdsetter mest.

Vi fant det fornuftig å ha sprint-økter på en uke med en Sprint Retrospective[7] (Schwaber, & Sutherland, 2008) hver fredag da vi jobbet ute hos MediaLT. Her kunne vi oppdage hva som gikk bra, gi forslag til forbedringer og tiltak som burde iverksettes før neste sprint. For å styre SCRUM-prosessen brukte vi en applikasjon som heter Trello. Her kan man opprette gjøremål på en tavle og flytte dem rundt etter hvor langt i prosessen man er kommet.

Vår veileder anbefalte oss å jobbe med programmeringen to og to i starten, hvor en skriver kode og den andre kommer med innspill. Dette fungerte veldig bra. Slik fikk vi lært de samme tingene og løst problemer mye lettere. Vi fortsatte med denne metoden stort sett gjennom hele prosjektet.

2.3 Kunnskapstilegning

Gruppen har brukt internett aktivt i denne oppgaven. Dette skyldes at det er lite informasjon om WAI-ARIA i bokform. Ting som allerede er utprøvd og testet dukker i stor grad opp i forskjellige webdesign-blogger og utviklingsmiljøer på nett, og har vært veldig nyttig for oss. Vi har gjennom nettblogger og diskusjonsforum kunnet sette oss inn i hva som er de normale problemstillingene ved mange av WAI-ARIA sine løsninger. WAI-ARIA sine offentlige nettsider gir mest grunnleggende info, men veldig lite om praktisk bruk. Det er også en oppfatning av at WAI opererer inni en lukket svart boks, og ikke har en like åpen diskusjon rundt standarden som de liker å tro. Derfor er det veldig mye informasjon og tips å hente fra webdesignere. Denne måten å hente informasjon på blir også mer utbredt i arbeidslivet. Det er uansett viktig å alltid holde en kritisk distanse til det som skrives på nettforum.

Rike applikasjoner utvikles veldig ofte i JavaScript, så dette har vi måttet lære oss. De fleste på gruppen hadde minimalt med erfaring, bortsett fra veldig enkle oppgaver fra diverse tidligere kurs ved HiOA. Vi deltok derfor på et JavaScript-kurs i regi av konsulentselskapet

[7] Sprint Retrospective; demonstrerer det man har laget for utvalgte interessenter og produkteier.

Bekk ("JavaScript - the good parts"), og benyttet Codecademy.com (Codecademy, 2012) som er et gratis online-kurs for JavaScript. To av gruppemedlemmene gjennomgikk i tillegg videokurset "30 Days to Learn jQuery" av Jeffrey Way (2012) etter behov. Dokumentasjonen til jQuery (The jQuery project, 2012) ble tatt i bruk for å forstå bruken av de forskjellige innebygde metodene.

Den sosiale spørsmål- og svar-siden, StackOverflow, har blitt tatt i bruk for å raskt finne gode programmeringsløsninger. Her kan man få hjelp fra profesjonelle programmerere og lese andres spørsmål og svar (Stack Exchange inc., 2012). Svarene har i tillegg ofte et preg av "best practice"-løsninger fordi de fingranskes av de andre brukerne og kan kommenteres og stemmes opp eller ned. Best practices er et nyttig for å enklest mulig kunne opprettholde tilgjengelighet på nett.

Alle gode lenker til temaet har vi lagret fortløpende og delt med hverandre gjennom Google Docs og også gjennom vår gruppeside på Facebook.

2.4 Kurs

Samspillet mellom kursene på HiOA har gitt oss et veldig godt utgangspunkt for å utføre vårt prosjekt.

Universell utforming (LO122D) har vært det mest nyttige faget i forhold til vår oppgave. Faget tok for seg lover og regler rundt universell utforming og rettningslinjene Web Content Accessibility Guidelines (WCAG) som er til stor hjelp under utvikling av tilgjengelige nettsider. Ved å følge WCAG kan man lett lage tilgjengelige nettsteder, og fikse tilgjengelighetsproblemer før man i det hele tatt går i gang med brukertestingen. Vi ser derfor på dette som en meget tidsbesparende ressurs. For å lettere kunne gjennomgå WCAG på vår nettside har vi brukt WebAIM's WCAG 2.0 Checklist (WebAIM, 2008) som

gjør det lettere forstå punktene i praksis. Der vi har ansett det som nødvendig har vi også hentet informasjon fra den offisielle WCAG-spesifikasjonen (W3C, 08.12.2008).

Kurset om databaser (LO149D) ga oss en god innføring i strukturering av data. Denne kunnskapen ble utnyttet i forhold til datasettet i en quiz vi utviklet - for eksempel har hvert spørsmål har et en-til-mange-forhold mot alternativene. Dette gjorde det lettere å både opprette og skalere datasettene etter våre krav.

Gjennom Web-prosjekt-kurset lærte vi også å samarbeide effektivt i en gruppe i tillegg til det grunnleggende av HTML og CSS.

Kunnskap om filoverføring ved hjelp av SFTP til skoleserveren, slik at nettstedet blir tilgjengelig på nett, har også blitt tilegnet gjennom kurs ved HiOA, blant andre Web-prosjekt (LO136A) og Webprogrammering (LO113D).

Hele gruppen har tatt kurset i Webprogrammering (LO113D) (med PHP) ved HiOA, og kunnskapene ble tatt i bruk ved utvikling av nettstedet.

I faget Systemutvikling (LO138D) fikk vi videreført kunnskapen fra Web-prosjekt og lærte mer om mer avansert prosjektstyring, prosjektorganisering og utviklingsmetoder - deriblant en introduksjon til SCRUM.

Menneske maskin interaksjon (LV125D) har nyttig i forhold til brukertesting av nettstedet.

Med kunnskaper fra Operativsystemer (LO114D) visste vi at vi kunne sette opp en symbolsk lenke slik at filene på Dropbox automatisk ble overført til den lokale installasjonen av webserveren WAMP.

2.5 Teknologiske verktøy og hjelpemidler

2.5.1 Planleggingsverktøy

Trello

Trello er en web-basert organiseringsapplikasjon som er formet som en oppgavetavle. Det fungerer ved at man bryter ned prosjektet i tasks i from av små oppgavekort med beskrivelser. Disse kortene kan man tilegne til en eller flere medlemmer av gruppen, man kan sette tidsfrist for oppgaven og man kan dele opp oppgaven i en liste med avkrysningsbokser og en progressbar. For hver nye oppgave opprettes et kort i en "To do-liste" som dermed overføres til en "Doing-liste" og ender i en "Done-liste".

Gruppen har brukt Trello som et Scrum-verktøy. Trello har det meste av funksjoner ved planlegging og gjennomføring innenfor Scrum-metoden som fordeling av arbeidsoppgaver, tidsfrister og status på oppgavene. Denne applikasjonen har gjort det lett for gruppen å holde seg oppdatert i prosjektet.

Google Docs

Google Docs (nå også kalt Google Drive) er en web-basert office-pakke i likhet med Microsoft Office. Docs består av diverse programmer;

- Document (fungerer i likhet med MS Word).
- Spreadsheet (fungerer i likhet med MS Excel)

- Presentation (fungerer i likhet med MS PowerPoint)
- Drawing er et bilderedigerings program
- Form er et program for å lage skjeamer som lagrer seg i Spreadsheet.

Alt brukeren lager i Docs blir lagret på brukerens Google-konto og kan deles med andre. Deler man et dokument med andre kan alle jobbe samtidig i dokumentet. Docs støtter de fleste vanlige fil-formatene, som for eksempel .doc, .xls, .ppt, .pdf osv, når man vil laste ned et dokument.

Gruppen brukte Docs Document til å skrive samtlige rapporter og dokumenter i prosjektet. Gruppen har også brukt Docs til å lage et skjema til brukerundersøkelsen gjennom Docs Form. Det ferdig skjemaet ble lagret med en egen link man kan sende ut til brukerne. Svarene ble lagret i Docs Spreadsheet hvor man har mulighet til å sammenlikne svar og lage grafer. (Grafer var uansett ikke så aktuelt for oss ettersom vi ble enige om 5 testpersoner). Docs har vært genialt i forhold til at gruppen lett kan samarbeide i samme dokument og at vi alltid har det vi har skrevet sikkerhetskopiert på nett.

Facebook

Facebook er et nettsamfunn hvor man kan lage en personlig brukerprofil, legge til venner og bekjente (andre brukerprofiler) og dele statusmeldinger og bilder. Mennesker med felles interesser innenfor for eksempel jobb, skole, hobby eller kjæledyr kan organisere seg i grupper og dele sin felles interesse med hverandre. Facebook har og en innebygget chat hvor man kan prate med sine påloggede venner eller sende meldinger til sine avloggede venner.

Gruppen opprettet en egen lukket facebook-gruppe, kalt “Tjueåtte”, for prosjektet og brukte denne som et hjelpemiddel for å sende hverandre litteratur, lenker til aktuelle hjelpesider, planlegge møter, dele informasjon og spørre hverandre spørsmål. I tillegg holdt vi jevnlig kontakt via Facebook sin chat.

Gruppenettside

På gruppens nettside førte vi prosjektdagbok og la ut alle dokumenter for offentligheten slik at andre kunne følge med hvor vi var i prosjektet. Dagboken ble oppdatert 2-3 ganger i uken med oppsummeringer av de dagene gruppen jobbet sammen. Det var åpent for å jobbe hjemme innimellom ettersom noen på gruppa har barn og andre har jobb. Vi syntes det ville vært fint å tilpasse nettstedet etter milde WCAG-krav slik at vi kunne ha et gjennomført universelt utformet prosjekt. Dette ble derimot gjort på siden av prosjektet og ble ikke vektlagt like tungt som det øvrige.

2.5.2 Utviklingsverktøy

Komodo Edit 7

Komodo Edit er en gratis teksteditor med støtte for autofullføring for blant annet HTML, CSS, PHP og JavaScript. Programmet finnes både til Windows, Mac og Linux, og gjør det dermed mulig for oss å dele felles erfaringer med bruken underveis i prosjektet.

Tre i gruppen har brukt Komodo som HTML-, CSS-, PHP-, og Javascript-editor. Den siste har brukt Dreamweaver CS4.

Sublime Text 2

Sublime Text 2 er en gratis teksteditor med de samme egenskapene som Komodo Edit 7 som er beskrevet over, men Sublime Text 2 har og også flere forskjellige plugins som gjør arbeidet lettere. En av de mest nyttige har vært automatisk FTP-opplastning, som betyr at de endringer man gjør i for eksempel HTML-filen vil automatisk lastes opp på nett når det lagres. Et annet nevneverdig tilleggsprogram er SublimeLinter som er en implementasjon av forskjellige lint-verktøy (JSHint med jQuery-støtte ble brukt) som sjekker og markerer fallgruver i forhold til koden.

Én har tatt i bruk Sublime Text for HTML-, CSS-, PHP-, og Javascript-utvikling.

FileZilla

FileZilla er et gratis FTP-program basert på åpen kildekode. Programmet finnes både til Windows, Mac og Linux.

Gruppen har brukt FileZilla til å laste opp filene til skoleserveren for å få nettstedet på nett.

Dropbox

Dropbox synkroniserer filer mellom PC-er, nettet og mobile enheter. Disse filene kan du gjøre tilgjengelig bare for deg selv, eller du kan dele dem med andre. Programmet lager en mappe (som standard i Mine Dokumenter) kalt "Dropbox". Her legger du filer du vil synkronisere, noe som gir mange muligheter:

1. Filen synkroniseres mellom to eller flere maskiner hvor du har Dropbox installert. Endringer du gjør på en fil lagret i Dropbox vil bli synkronisert over til en annen maskin hvor man kan fortsette på arbeidet.
2. Filen synkroniseres til dropbox.com. Man har da tilgang til fila hvor enn det er internettilgang. F.eks. kan man laste ned Powerpoint-presentasjonen man skal bruke fremfor å dra rundt på en USB-pinne.
3. I Dropbox kan du opprette en undermappe og dele med venner eller kollegaer. Man kan da arbeide sammen på et prosjekt og endringer vil synkroniseres i sanntid, eller man kan dele filer med hverandre (feriebilder, en kul sang etc.)
4. Hvis datamaskinen har krasjet og filene har forsvunnet kan man installere Dropbox på den nye maskinen og alle filene vil bli lastet ned igjen. Man kan også opprette tidligere filer (slettet ved uhell), eller tidligere versjoner av samme fil (skulle den plutselig bli korrupt). Denne sikkerhetskopifunksjonen er kjekk å ha for fotografier og andre uvurderlige filer.

Dropbox har vært det viktigste verktøyet for Gruppe 28 i forhold til sikkerhetskopi, deling og tilgjengelighet av filer.

WampServer

WampServer er et webutviklingsmiljø til Windows som gjør det enkelt å teste PHP-kode lokalt. Programmet består av Apache, PHP og MySQL som til sammen utgjør en fullt funksjonell webserver. Ved å bruke symbolske lenker (en slags speiling av filene) har vi fått dette til å fungere automagisk med Dropbox, slik at vi ikke trenger å flytte filer manuelt.

Gruppen har brukt WampServer til å teste PHP-filer under utviklingen.

2.5.3 Nettlesere og nettlesertillegg (analyseverktøy)

Internet Explorer 9

Internet Explorer 9 (IE9) er den siste versionen til Microsoft sin standard-nettleser. IE9 sies å være den sikreste nettleseren på markedet (i følge Microsoft selv). IE9 har støtte for CSS3 og HTML5.

Én i gruppen bruker bruker IE9 som standard nettleser, ellers er det bare brukt for å teste demoene.

Google Chrome 19

Chrome er Google sin nettleser og sies å være den raskeste ettersom hver fane i nettleseren kjøres som en enhet i prosessoren. I tillegg er Chrome populær fordi man kan legge til apps, små programmer og spill man kan kjøre i nettleseren. Chrome har støtte for CSS3 og HTML5.

Én i gruppen bruker Chrome som standard nettleser, ellers er det bare brukt for å teste demoene.

Mozilla Firefox v10-12

Firefox er en open source nettleser fra Mozilla. Firefox er populær for alle sine tilpasningsmuligheter, man kan egendefinere den til sin egen stil og legge til masse

funksjonalitet gjennom små tilleggsprogrammer kalt addons. Firefox har støtte for CSS3 og HTML5.

To i gruppen bruker Firefox som sin standard nettleser, ellers er det bare brukt for å teste demoene.

Opera v11.64

Opera er en gratis nettleser laget av det norske selskapet Opera Software. Nettleseren har i likhet med Firefox og Chrome addons som de kaller extensions og apps som de kaller widgets, men ikke et like stort utvalg. Opera var kjent for å være tidlig ute med støtte for HTML5 og CSS3.

Gruppen har brukt Opera bare for å teste demoene.

Safari v5.1.7

Safari er standard nettleseren til Mac, men kan også lastes ned til Windows. Safari er den eneste nettleseren på Mac som gir støtte for VoiceOver, Mac sin innebygde skjermleser. Safari er og kjent for å være rask og ha innebygd staveretting. Safari har støtte for CSS3 og HTML5.

En på gruppen bruker regelmessig Safari i sammenheng med VoiceOver for å teste demoene.

Juicy Studio Accessibility Toolbar 1.7

Dette tillegget til Firefox gjør det mulig å se gjennom WAI-ARIA-attributter på en nettside. JSAT har også mulighet for å sjekke fargekontrasten til siden.

Gruppen har brukt dette for å se landemerker, live-regioner og for å teste fargekontraster på nettstedet.

WAVE Toolbar

Et tillegg til Firefox som analyserer og graderer nettstedets tilgjengelighet. Man får opp feilmeldinger ved problemer med tilgjengelighet og andre ting som ikke er riktige. I tillegg viser Wave hvor det er blitt brukt WAI-ARIA.

Gruppen har brukt Wave for å finne tilgjengelighetsfeil på nettstedet.

Firebug og Chrome Developer Tools

Disse tilleggene fungerer på samme måte til Firefox og Chrome. De gjør det mulig utføre feilsøking ("debugging") i JavaScript-filer og har vært en uvurderlig ressurs under utviklingen av våre rike applikasjoner.

Disse har blitt brukt for å se på koden til diverse nettsider med funksjoner av interesse ,og for å teste endringer på design og egne funksjoner uten å trenge å gå inn i selve filene.

FireQuery

Et tillegg til Firebug som gjør det mulig å jobbe med jQuery-objekter. Gruppen har brukt det for å enklere jobbe med jQuery-objekter i Firebug.

WCAG Contrast Checker

Dette er en addon til nettleseren Firefox og brukes til å sjekke kontrastnivået av farger på en nettside i forhold til WCAG-standarden.

Gruppen har brukt denne i gjennom hele nettstedet for å få all kontrast i tråd med WCAG sin standard.

WorldSpace FireEyes

FireEyes er et tillegg til Firefox som gjør det mulig å utføre avanserte tester i forhold til WCAG 1.0 og 2.0. Fordelen med dette programmet er at det både kan analysere statisk og dynamisk innhold (for eksempel JavaScript).

Gruppen har brukt dette teste quizen som er laget med blant annet jQuery.

DOM Monster

DOM Monster er et JavaScript-bokmerke til nettlesere som rapporterer problemer i forhold til bruken av DOM-elementer på en webside. Det gir også tips til forbedringer.

Gruppen har brukt DOM Monster på flere av demoene som er kodet med javaScript for å få tips til å bedre nettsidenes respons-tid.

Fangs Screen Reader Emulator

Fangs er et tillegg til Firefox som skriver ut det JAWS eller andre skjermlesere hadde lest opp.

Fangs har vært brukt for å utføre kjappe tester der det ikke har vært nødvendig med JAWS eller NVDA.

JAWS 13

JAWS er den mest populære skjermleseren på markedet i dag (Freedom Scientific, INC, 2012). Det er derfor hensiktsmessig å utføre testing med JAWS fordi man dekker den største markedsandelen for den blinde og svaksynte målgruppen. Baksiden er at gratisversjonen av programmet bare kan kjøres i 40 minutter for hver gang datamaskinen startes på nytt, og fullversjonen koster over 800 dollar. Vi har også vært usikre på om demolisensen er lov å bruke i test- og utviklingssammenheng.

Gruppen har for det meste forholdt seg til NVDA men har brukt JAWS for å teste kompatibiliteten til demoene.

NonVisual Desktop Access (NVDA)

NVDA er en gratis skjermleser basert på åpen kildekode. NVDA fungerer bare til Windows og tilbyr tilbakemelding via syntetisk tale og braille. NVDA er et verktøy som lar blinde og

svaksynte bruke PC på tilnærmet samme måte som seende. NVDA har støtte for 35 språk, inkludert norsk, og kan kjøres rett fra en USB-enhet uten installasjon (NV Access, 2011).

Gruppen har hovedsakelig valgt å utføre vår testing ved hjelp av dette verktøyet fordi det er enkelt å ta i bruk, gratis og har støtte for WAI-ARIA.

2.5.4 Programmeringsspråk og rammeverk

HTML5

HTML 5 er den femte versjonen av kodespråket HTML, som brukes for å presentere innhold på web. HTML5 er fortsatt under utvikling, men vil etterhvert ta over etter forrige versjon, HTML 4.01. Nettlesere har litt forskjellig støtte når det kommer til HTML 5, og ingen har ennå støtte for alle funksjonene HTML 5 tilbyr (W3Schools, 2012a).

Gruppen ble enig med MediaLT om å ta i bruk HTML5 i vårt prosjekt fordi det tilbyr nye og bedre løsninger på tilgjengelighetsproblemer innenfor universell utforming.

CSS3

CSS3 er neste generasjons CSS(Cascading Style Sheets) versjon etter CSS2 og er fortsatt under utvikling (W3Schools, 2012b). CSS lar oss skille designet fra innholdet på nettsider.

Gruppen har brukt CSS for å designe nettstedet og gjøre den mer brukervennlig i forhold til fargevalg og oppbygning.

JavaScript

JavaScript er et scriptspråk som støttes av alle de store nettleserne på markedet i dag. Scriptene blir vanligvis kjørt på brukerens nettleser slik at det gir mulighet for responsive interaktive funksjoner.

Gruppen har tatt i bruk JavaScript fordi det er det fundamentale verktøyet ved utvikling av rike webapplikasjoner.

jQuery

jQuery er et rammeverk til JavaScript som gjør det enklere å hente ut og sette inn elementer på en webside.

Gruppen har valgt å bruke jQuery på noen av demoene ettersom det har bedre og enklere løsninger for å sette inn elementer og endringer.

jQuery UI

jQuery UI er et ferdig bibliotek til jQuery som har en rekke UI-løsninger som gjør det mulig opprette ferdig skreddersydde elementer med med både design og attributter.

Gruppen har brukt jQuery UI for å sammenligne koden det generer for å se om det kunne gjøres forbedringer på elementer som ble laget manuelt. I vår situasjon ville det ikke lønnet seg å selv bruke jQuery UI ettersom det ville blitt vanskeligere å se oppbygningen for noen som vil prøve å lage de samme demoene vi eventuelt hadde brukt dette på.

PHP

PHP er et er et scriptkodespråk som er tilpasset HTML og er mest brukt for å lage dynamiske nettsider.

Gruppen har brukt PHP for å lage maler og for å hente frem designet og brukergrensesnittet på hver av de enkelte nettsidene inn på nettstedet.

Google-Code-Prettify

Google-Code-Pretty er en funksjon for å markere og utheve kildekode-syntaks gjennom CSS ved hjelp av et JavaScript-bibliotek.

Gruppen har brukt Google-Code-Prettify på kodeboksene for å sette fargekode på den fremviste koden i boksen.

2.5.5 Andre verktøy

Photoshop CS4

Vi har brukt Photoshop for å lage våre egne illustrasjoner til nettstedet. Vi har også designet våre egne prosjekt-maskoter til gruppenettsiden og alt av prosjekt-materiale.

3. Mål og krav

3.1 Oppdraget

Oppdraget vårt gikk ut på å bidra til MediaLT sitt bokprosjekt ved å lage et supplement til boken. Supplementet skulle være i form av en nettside hvor gruppen skulle ta for seg W3Cs nye universelle utformingsstandard WAI-ARIA. Ettersom ARIA er veldig nytt og ennå lite brukt vil det være svært nyttig å ha norsk læremateriale på dette området. Dette vil bidra til å gi mer fokus og kjennskap til WAI-ARIA i det norske utviklermiljøet. I tillegg skal alle eksisterende IKT-løsninger som underbygger virksomhetens alminnelige funksjoner, og som er hovedløsninger rettet mot eller stillet til rådighet for allmennheten være universelt utformet fra 1. januar 2021 og alle nye løsninger fra 1. juli 2011(BLD (Barne- likestillings- og inkluderingsdepartementet), 2012). Dette vil dermed kunne øke interessen for dette feltet fremover.

3.2 WAI-ARIA

Web Accessibility Initiative (WAI) er en del av W3C, og er et sett med protokoller for økt tilgjengelighet på nett. W3C står bak to av de viktigste web-standardene vi kjenner i dag, nemlig HTML og CSS. Accessible Rich Internet Applications Suite (ARIA) er en del av WAI, og omtales derfor som WAI-ARIA. WAI-ARIA benyttes som et tillegg til HTML for universell presentasjon og er spesielt nyttig for dynamisk innhold.

3.2.1 Rike applikasjoner

Dagens nettsider oversvømmes ofte av visuelle elementer i form av små webapplikasjoner. JavaScript er en teknologi som støttes av de aller fleste nettlesere på både mobile og

stasjonære operativsystemer. Denne teknologien er ofte ikke godt tilpasset for funksjonshemmede. Den økte bruken av JavaScript, JQuery, Dojo, Flash og andre kodespråk / toolkits gir en mer uforutsigbar opplevelse enn tidligere. Når flere av nettstedets viktigste funksjoner baserer seg på disse webapplikasjonene er det nødvendig å sørge for at vi alle får tilgang. Dynamisk innhold kan for veldig mange oppleves som forvirrende og navigering kan bli ekstra tungvint. Ettersom en stor del av befolkningen avhenger av **assisterende teknologi** og benytter tastaturnavigasjon isteden for mus, kan disse brukerne ende opp med å bli ekskludert. Hvordan skal man gjøre drag and drop, varselsbokser og visuelle oppdateringer tilgjengelig for disse brukerne? For å løse dette kan man benytte WAI-ARIA.

3.2.2 Hvordan bruke WAI-ARIA

WAI-ARIA bidrar til å bryte barrierene for synshemmede. Standarden setter fokus på rike applikasjoner og gir informasjon om hvordan å lage logisk struktur på nettsider. Man kan tenke på ARIA som en tolk mellom nettleseren og de assisterende hjelpemidlene. Slik sørger man for at handlinger får et formål og også forutsigbare konsekvenser.

For å gi brukerne ekstra informasjon om elementer på en nettside benytter ARIA en **role**-klassifisering. Disse rollene tjener til å støtte navigering og gir oversikt over siden, samtidig som man kan identifisere widgets (dynamisk innhold). Slik kan man informere om at man nå går fra en statisk nettside til en applikasjon (ved å definere området med role="application").

Alle HTML-elementer kan i tillegg tilordnes **ARIA-attributter**. Under hver ARIA-atributt kan man legge til **tilstander og egenskaper** for hvert element. Dette gir ekstra

opplysninger om hvert element, som f.eks. om en avkrysningsboks er avhuket, og også om avhuking er påkrevd. En egenskap kan være *ARIA-labelledby* som er vedvarende. En egenskap som *ARIA-pressed* kan derimot endres til *="true" eller ="false"*. Endringer i tilstander og egenskaper gjøres typisk ved hjelp av JavaScript og da oftest på denne måten: *<element>.setAttribute("attributtnavn", verdi).* På elementer hvor man åpner for interaksjon med mus bør man også ha med muligheter for tastaturhåndtering.

ARIA-standarden er ment som et supplement, og man bør bruke metoder for tilgjengelighet hvor det allerede eksisterer. Det langsiktige målet er nemlig å gjøre ARIA overflødig og kunne tilby universell utforming fra grunnen av.

3.3 Oppgaven

NB: Dypere forklaring av dette innholdet vil du finne under "Produktdokumentasjon". Oppgaven gikk ut på å lage demonstrasjoner på bruk av WAI-ARIA og presentere dette i form av en nettside kodet i HTML5. Gruppen skulle ta for seg fire hovedkategorier innenfor WAI-ARIA:

1. **Landemerker (Landmarks)**. Ved bruk av landemerker vil et HTML-dokument deles opp i forskjellige logiske ARIA-roller. Dette kan være elementer som header, hovedinnhold, menyer og bunntekst som allerede separeres logisk ved hjelp av CSS. Landemerker tydeliggjør disse elementene for assisterende teknologi.

2. **Tabindex** settes for å navigere seg lett igjennom en HTML-side med tabulatorknappen. Tabindex brukes for å sette rekkefølge på hvordan tabulatoren beveger seg gjennom innholdet av siden. Det kan også brukes til å gjøre det mulig å fokusere på elementer som ikke vanligvis kan fokuseres på (for eksempel innholdstekst i p-tags).

3. **Live-regioner (Liveregions)** brukes for innhold som oppdateres dynamisk, og gir brukeren mulighet til å definere graden av hvor ofte og hvor mye man ønsker brukeren skal få opplest via sin skjermleser.

4. **Widgets** er små applikasjoner som kalkulator, besøksteller, klokke o.l. som kan ha en uventet oppførsel på skjermlesere avhengig av hvordan de er programmert. ARIA har flere løsninger på hvordan man kan gjøre disse tilgjengelige.

I tillegg kunne vi lage sammensatte applikasjoner som spill o.l. for å illustrere forskjellig ARIA-funksjonalitet. Det ble gitt noen forslag på dette men oppdragsgiver ga ellers ganske frie tøyler her slik at vi kunne få være litt kreative.

3.4 Kravspesifikasjonen og dens rolle

Kravspesifikasjonen er ment som en veileder for at både gruppen og oppdragsgiver skal ha samme forståelse av prosjektet, og sikre at sluttresultatet blir i tråd med alles forventninger. Hensikten er også at vår prosjektveileder skal kunne bruke kravspesifikasjonene som en rettesnor når gruppearbeidet blir evaluert fortløpende.

3.4.1 Kravspesifikasjonen

Disse kravene ble satt for utviklingen av nettstedet. Nettstedet skal fungere som en guide til WAI-ARIA-funksjoner med forklaring på hva funksjonen er, en demonstrasjon på den, samt kodeforklaring. Målgruppen til nettstedet vil være webutviklere/designere og IT-studenter. Ved utvikling av demoer tar vi utgangspunkt i forslag vi har fått av vår oppdragsgiver i MediaLT, Morten Tollefsen. Men vi fikk ellers få konkrete rammer, så kravspesifikasjonen utarbeidet vi ut i fra de løse kriteriene som ble gitt.

Nettstedet

- Skal være en brukervennlig guide til WAI-ARIA.
- Skal være utviklet i HTML 5 og CSS3.

- Skal være utformet etter WCAG 2.0 standard.
- Skal ta i bruk WAI-ARIA-elementer der det lar seg øke brukervennlighet.
- Toppmeny skal bestå av generell informasjon.
- Sidemeny skal bestå av demo-forklaringer, demoer, og eksempler.
- Skal være oppdelt via HTML- og PHP-maler for å slippe å måtte føre inn menyer osv. (relativt statisk innhold) på alle nye filer.
- Det estetiske designet på nettstedet skal ikke være overdrevet, men skal ha fokus på oppbygning med en logisk struktur.

Demo

- Skal ta for seg fire hovedkategorier innenfor WAI-ARIA
    - Landemerker
    - Live-Regioner
    - Widgets (Rike applikasjoner)
    - Tabindex
- Skal lage universelt utformede spill i form av rike applikasjoner.
- Skal utvikles i HTML, JavaScript/jQuery og/eller DOJO.
- Skal være lett å forstå og ha enkle forklaringer på hva ting er og hvordan det fungerer.
- Skal ha kodebokser med fargekoder på koden som presenteres. Gjøres ved hjelp av Google-Code-Prettify.

3.4.2 Endringer i Kravspesifikasjonen

Gruppen droppet å utvikle demoer i kodespråket DOJO etter som vi hadde nok å lage i javaScript og jQuery, og det ville gått mye tid på å lære seg ennå et nytt kodespråk.

Det ble gitt noen tidlige forslag til spill og quiz som vi kunne utforske, men dette ble ikke låst til noe krav så her har vi gjort noen små endringer, bl.a. å lage en ARIA-quiz istedet for hovedstadsquiz, og et klikkespill.

I tillegg er hver demonstrasjon testet med et utvalg nett- og skjermlesere og resultatet notert på siden under hver demonstrasjon. Dette var i utgangspunktet ikke et krav, men vi mente det var nødvendig å ha med dette da vi innså at det var en del forskjellig oppførsel.

3.4.3 Bruk av kravspesifikasjon

Siden er laget med HTML5 og CSS3 i henhold til kravene. Designet på siden er testet med 5 testpersoner hvorav den ene er blind og har testet med skjermleser. Her ble det gjort en del endringer. Fremvisningen av kode var utydelig i starten men ble endret til bedre farger. Siden oppfyller ellers WCAG 2.0. Språket på nettstedet har vært i stadig endring for å gjøre det mer pedagogisk. Dette har også blitt endret etter brukertesting.

Det er også utformet maler i PHP og HTML som man kan benytte ved utvidelse av siden.

4. Om utviklingsprosessen

4.1 Valg av formidling

Oppdraget vårt var å lage demonstrasjoner på god ARIA-praksis og finne gode løsninger på vanlige tilgjengelighetsproblemer i rike applikasjoner. I utgangspunktet var ikke oppdragsgiveren spesielt interessert i presentasjon, design og CSS. Vi kunne dermed valgt å lagd en samling enkle HTML-sider uten noe mer rundt. Men med tanke på at dette prosjektet var temmelig åpent, og at målgruppen er webdesignere og studenter, valgte vi å presentere kodene våre på en lærerik måte på en nettside som var klar til bruk. Slik kunne vi også få utforsket WCAG sammen med ARIA. Dette ble dermed en del av vår kravspesifikasjon.

I vårt tilfelle er ikke oppdragsgiver i stand til å bedømme vår CSS, men kun nettsidenes oppbygning. Ved å presentere prosjektet vårt for oppdragsgiveren underveis har vi fått en god forståelse for hvordan en nettside fremstår for blinde.

4.2 Tilpasninger til målgruppen

I dette prosjektet har det vært viktig å holde fokus på hvem som er målgruppen, nemlig studenter som ønsker å lære bruken av ARIA. Denne gruppen inkluderer også blinde, svaksynte, dyslektikere osv. som vi vil måtte brukerteste for. Det ville også vært svært selvmotsigende om vi ikke praktiserte våre egne metoder i selve nettstedets struktur. Det er vårt mål at nettstedet skal være så brukervennlig som mulig etter gjeldende standarder som WCAG og ARIA. Vi har underveis i prosjektet fått en del utfordringer med hensyn til dette.

En god nettside bør helst være konsekvent med faste menyer. Skal man derimot gi et god demonstrasjon av f.eks. bruken av tabindex ville dette resultert i en kollisjon mellom de allerede eksisterende tabfunksjoner (i faste menyer) og det aktuelle testområdet. Derfor er det mer hensiktsmessig å fjerne alt rundt som kan forstyrre tabsekvensene vi ønsker å vise, slik at man kan få bedre utbytte av selve demonstrasjonen og slippe å passere hele menyen hver gang.

Med dette i tankene ble det besluttet å fjerne hoved- og sidemenyen på hver av demonstrasjonene. Aller helst ønsket vi å ha en helt enkel demonstrasjon på hver av tingene slik at man ved testing av skjermleser ikke ble forstyrret av alle andre beskjeder. For den gruppen som leser nettsider på vanlig måte og som også er uerfarne ved bruk av skjermlesere, vil de faste menyene derfor gjøre demonstrasjonen ekstra kompliserte.

På den andre siden vil dette igjen komplisere strukturen for den gruppen som avhenger av skjermleser, i det at de vil miste sine naturlige holdepunkter. For å gjøre det enklest for begge brukergrupper har vi derfor hatt med kun et minimum med nødvendigheter på demo-sidene.

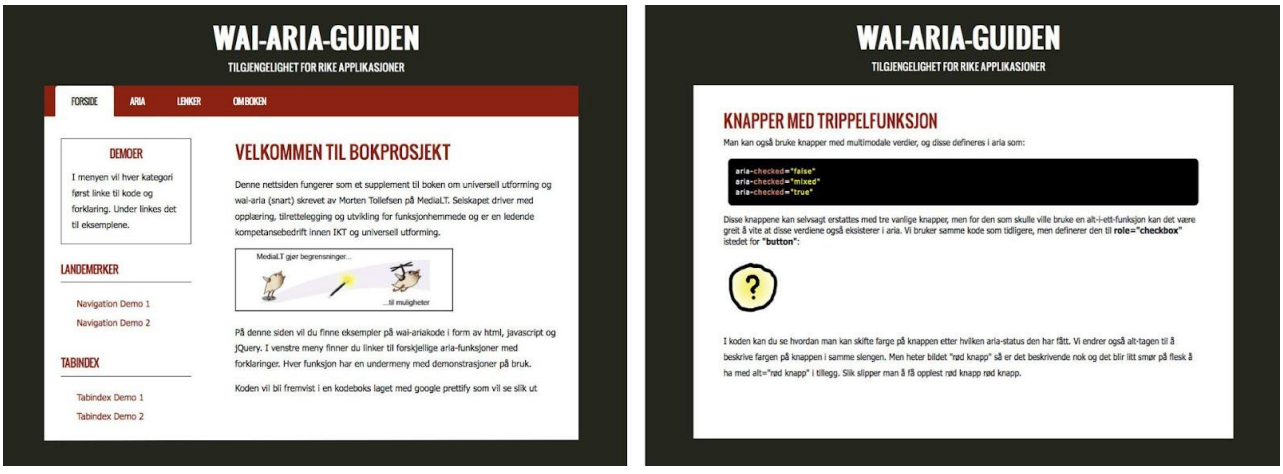


Fig1. Her vises forsiden og en demo-side, hvor man kan se hvordan layouten er gjort enklere på eksempelet til høyre.

Headeren er konsekvent med på alle sider, og det er en *tilbake til index*-link og en *neste*-link på bunnen av hver demo. Resten er fjernet. Ved hjelp av CSS har vi beholdt den samme designen på demonstrasjonene slik at man skjønner at man er på samme nettside.

4.3 Andre utfordringer

4.3.1. Kollisjon med skjermleser

Ved utviklingen av lydspillet startet vi med å programmere et universelt utformet spill. Spillet baserer seg på lydgjenkjenning og man trenger kun å navigere seg gjennom knapper og kunne trykke på dem. Så snart man har funnet to og to like vil disse deaktiveres og man får en ny boing-lyd på disse som indikerer at knappene nå er deaktivert. Slik blir det ikke nødvendig med ARIA-atributter for å fortelle at knappene er valgt. Dette er i grunnen en veldig brukervennlig løsning i utgangspunktet.

Når skjermleseren skrus på, oppstår det derimot noen underlige effekter, og de er underligere hvis ARIA brukes enn hvis det ikke er i bruk. Vi fant ut at her kunne det være fint å vise fram dette, ved å la halvparten av knappene være ARIA-kodet, den andre halvparten ikke. I de to øverste radene har knappene ikke ARIA-koding, og skjermleseren (NVDA) leser opp posisjon og tilstand på denne formen ved hhv. ikke funnet og funnet knapp:

"Ikke funnet 4 av 4 rad 2 kolonne 4. Ikke funnet knapp."

"Funnet 4 av 4 rad 2 kolonne 4. Funnet knapp"

Ikke like bra som det burde være, men forståelig, i hvert fall etter noe tilvenning.
Med ARIA-koding, i de nederste radene, leses det opp slik:

"Ikke funnet 1 av 4 rad 3 kolonne 1. Ikke funnet trykknapp ikke avkrysset."

"Funnet 1 av 4 rad 3 kolonne 1. Funnet trykknapp trykket ikke avkrysset"

Her blir det en del smør på flesk. Verst er det imidlertid at ARIA-kodingen gir tilsynelatende selvmotsigende opplysninger; *"Funnet trykknapp trykket **ikke avkrysset**".*

Det kan se ut til at skjermleseren pga. ARIA-kodingen bruker verdien til attributtet "aria-checked", som ikke er i bruk i dette eksempelet. Her nevner vi at dette attributtet ifølge ARIA ikke skal brukes på "button"-elementer[8], så dette problemet kan være en feil i NVDA.

Knappene ble satt i en tabell og merket med **table role="grid"**, dette for å kunne få til en bedre navigering rundt knappene. W3C beskriver bl.a. funksjonen slik: "Grids allow the user to move focus between cells using two dimensional navigation".[9] Dette ble likevel forvirrende for skjermleseren JAWS som slet med å holde fokus, og ikke fant ut hvilke knapper som var inaktive.

Den viktigste lærdommen av dette er at ARIA-bruk ikke alltid gir større tilgjengelighet eller brukervennlighet, og at bruken må testes og vurderes nøye i hvert enkelt tilfelle.

4.3.2. Øvrig testing

Ved testing av demonstrasjonene opplevde vi veldig varierende oppførsel fra skjermlesere og nettlesere. Enkelte nettlesere har implementert støtte for HTML5 mens andre henger etter, og noen skjermlesere støtter ARIA og andre ikke. I tillegg må man ta høyde for at skjermlesere og nettlesere kan ha personlige innstillinger som kan gi uventet oppførsel.

[8] http://www.w3.org/TR/wai-aria/roles#button

[9] **http://www.w3.org/TR/wai-aria/roles#grid**

5. Avslutning

5.1 Videre utvikling

Nettstedet skal tjene som utgangspunkt for en ressursside til en bok om universell utforming og ARIA. Målet er å lage et grunnlag slik at man enkelt skal kunne bygge videre på siden ved å legge til flere demonstrasjoner i tråd med hva som skrives i boken. Det vil selvsagt bli nødvendig å endre tekst, meny og overskrifter i samsvar med kapitler i boken.

Siden er laget med PHP-includes slik at man ikke trenge å bruke tid på WCAG-vennlig layout. For enkel videreutvikling har vi laget en *mal.php* med ferdig layout for hovedside med basisinformasjon om hvert tema, og en *maldemo.html* for hver enkle demonstrasjon. Ved hjelp av disse kan man enkelt utvide nettstedet ved å lime inn tekst og kode ved hjelp av prettyprint-kodebokser og øvrige menyer. Det er også fullt mulig å legge til flere spørsmål i quizen og flere lyder og knapper i lydspillet.

Vi ønsker at vår løsning skal bidra til mer oppmerksomhet og fokus rundt tilpasning av rike applikasjoner på nett og praktisk bruk av ARIA. Vårt mål er at ressurssiden vil være til nytte for norske webutviklere og studenter i faget "Universell utforming" ved høyskoler og universiteter i Norge.

Veldig nylig fikk vår oppdragsgiver Morten Tollefsen bekreftet at boken om universell utforming skulle realiseres. Dette høyner dermed sjansen for at Morten vil kunne bruke vårt prosjektresultat i sitt videre arbeid.

5.2 Eget utbytte

Ved utviklingen av denne siden har vi fått økt fokus på tilgjengelighet og har lært viktigheten av å skille innhold, struktur og utseende, både i HTML og i rike applikasjoner. Vi vil nå være i stand til å belyse vanlige tilgjengelighetsproblemer som normalt ikke

vektlegges og vi har også fått økt innsyn i hvordan en nettside fremstår for blinde. Vi antar at vi kommer til å ha god bruk for dette i arbeidslivet, ikke minst med tanke på diskriminerings- og tilgjengelighetslovgivningen for IKT som allerede har startet å tre i kraft.

Dette prosjektet har vært spesielt gunstig for oss på Anvendt Datateknologi ettersom studiet delvis er av samfunnsfaglig art. Vi har i dette prosjektet fått brukt vår kunnskap tilegnet på HiOA. Vi er også fornøyde med å sitte igjen med et produkt som kan brukes av andre studenter. Resultatet vårt er så langt vi kan se Norges første norske nettside som fungerer som en guide til WAI-ARIA.

5.2.1 Hva vi kunne gjort annerledes

Alle på gruppa var enige om at hadde vi startet forfra så ville vi nok brukt mindre tid på det visuelle og heller utforsket ARIA enda grundigere. Det største vendepunktet i prosjektet kom forøvrig da vi skulle ha prøvefremføring og tilskuerne poengterte at PowerPointen vår ikke var like tilgjengelig og leselig som de prinsippene vi fremmet. Her innså vi at vi måtte forbedre tilgjengeligheten på alt vi leverte, skulle vi fungere som de ambassadørene for tilgjengelighet som vi ønsket å være. Derfor vektla vi dette litt tyngre enn vi opprinnelig hadde planlagt. Vi har nå fått god erfaring på to områder istedet for ett, som alt i alt gir en litt bredere kompetanse. Man kan nemlig ikke være ekspert på ARIA uten å beherske de grunnleggende tilgjengelighetsprinsippene. Vår oppdragsgiver har også alltid vært klar på at vi stod fritt i dette prosjektet til å utføre oppdraget etter vårt eget ønske, og hadde få innvendinger på retningen prosjektet tok.

Ved skriving av kode kunne vi gjerne vært enda tydeligere i kommenteringen slik at man ved gjenbruk av filene lettere kan foreta endringer.

5.3 Konklusjon

Fremdriften av prosjektet har gått bra. Gruppen har samarbeidet godt og hatt god kommunikasjon seg i mellom. Gruppen fikk en litt sen start med selve utviklingen ettersom det var en del å sette seg inn i forhold til å lære seg WAI-ARIA og JavaScript og litt mye tid ble brukt på å lage et design til nettstedet.

Utviklingen i SCRUM fungerte godt. Dette var den eneste utviklingsmetoden som appellerte til oss og virket logisk i forhold til vårt utviklingsprosjekt. Noen komplikasjoner var det i forhold til å opprettholde hyppige korte møter og forholde seg til fastsatte utviklingsplaner, men det kom av at gruppen var litt uerfaren. Ved å bruke tjenesten TRELLO som et SCRUM-verktøy fikk gruppen organisert og planlagt prosjektet både enklere og bedre.

Gruppen føler de har fått tatt i bruk mye av lærdommen de har opparbeidet seg gjennom tiden på HiOA og dette har kommet godt med i prosjektet, men gruppen måtte også tilegne seg annen kunnskap selv om JavaScript, jQuery og WAI-ARIA. JavaScript ble lært igjennom internett og et dagskurs, jQuery ble lært gjennom internett og WAI-ARIA ble lært igjennom gruppens veileder Morten Tollefsen og internett.

Gruppen har hatt en fin opplevelse med å jobbe med MediaLT og Morten Tollefsen. Med tanke på at dette var et åpent prosjekt føler gruppen at de ikke bare har bidratt med dette for MediaLT sin grunn, men og for å fremme tilgjengeligheten av WAI-ARIA og universell utforming for alle som har lyst til å lære. Vi er hvertfall sikre på at vi nå sitter på verdifull kompetanse på dette feltet som kan vise seg å være attraktivt for fremtidens utvikling.

Produktdokumentasjon

"Internetts styrke er at det er universelt. Det er helt grunnleggende at alle mennesker har tilgang. Dette gjelder uansett hvilke funksjonsnedsettelser de måtte ha."

Tim Berners-Lee

Forord

Denne produktdokumentasjonen skal gi en forståelse av hvem produktet er rettet mot, hvorfor det er laget og hvordan det bør benyttes. Den er ment for alle som ønsker å ta siden i bruk, og som en dypere forklaring på problemstillingene som presenteres. I tillegg har vi med designbeskrivelse for nettsiden, og forklaring på hvorfor siden oppfyller W3C sine designkrav om god universell utforming.

Dokumentasjonen er delt inn i følgende avsnitt:

- **Kravspesifikasjon**, hvilke krav nettsiden skal oppfylle
- **Beskrivelse av Produktet**, hva er målet med nettsiden
- **Nettsidens oppbygging**, hvordan er nettsiden designet
- **Beskrivelse av demoer**, utfyllende om koden bak hver demo
- **Avslutning og konklusjon**, fremtidige muligheter for nettsiden

Innholdsfortegnelse

1. Kravspesifikasjonen

Disse kravene ble satt for utviklingen av nettsiden. Ved utvikling av demoer her vi tatt utgangspunkt i forslag vi har fått av vår kontaktperson i MediaLT, Morten Tollefsen. Men vi har hatt frie tøyler til å utforme demoene som vi selv ønsker.

Nettsiden

- kal være en brukervennlig guide til WAI-ARIA.
- Skal være utviklet i HTML 5 og CSS3.
- Skal være utformet etter WCAG 2.0 standard.
- Skal ta i bruk WAI-ARIA-elementer der det lar seg øke brukervennlighet.
- Toppmeny skal bestå av generell informasjon.
- Sidemeny skal bestå av demo-forklaringer, demoer, og eksempler.
- Skal være oppdelt via html- og php-maler for å slippe å måtte føre inn Menyer osv. på alle nye filer.
- Det estetiske designet på nettsiden skal være nøkternt og fokusere på oppbygning med en logisk struktur.

Demo

- Skal ta for seg fire hovedkategorier innenfor WAI-ARIA
    - Landemerker
    - Live-Regioner
    - Widgets (Rike applikasjoner)
    - Tabindex
- Skal lage universelt utformede spill i form av rike applikasjoner.
- Skal utvikles i HTML, JavaScript/jQuery og/eller DOJO.
- Skal være lett å forstå og ha enkle forklaringer på hva ting er og hvordan det fungerer.
- Skal ha kodebokser med fargekoder på koden som presenteres. Gjøres ved hjelp av Google-Code-Prettify.

1.1 Endringer i Kravspesifikasjonen

Gruppen droppet å utvikle demoer i kodespråket DOJO ettersom vi fant på nok å lage i JavaScript og jQuery, og det ville gått mye tid på å lære seg ennå et nytt kodespråk.

Opprinnelig ble det foreslått å ha med en ARIA-vennlig quiz hvor man for eksempel gjetter hovedsteder, men vi valgte å lage en quiz med spørsmål fra ARIA for at brukerne kunne teste seg selv etter å ha lest om funksjonaliteten. Oppdragsgiver syns at dette var en veldig god idé.

2. Beskrivelse av produktet

2.1 Tjenestens funksjonalitet

Vi har valgt å kalle nettsiden WAI-ARIA-guiden. Den skal fungere som en brukervennlig nettside for studenter, webdesignere og andre som ønsker å lære noe om ARIA. Vi har forsøkt å gjøre siden lærerik, intuitiv og litt morsom slik at man kan hygge seg mens man lærer. Siden tar for seg de grunnleggende prinsippene i ARIA med eksempler man kan teste ut selv, og informativ tekst om hvert problemområde, forslag til løsninger på problem, samt kode man kan kopiere. Under hver kategori gir vi en innføring i hvorfor ARIA er nødvendig og i hvilke tilfeller man bør vurdere å bruke det.

Siden kan i tillegg til å demonstrere ARIA også gi et eksempel på god praksis i utforming av tilgjengelige nettsider. Dette inkluderer god kontrastbruk, logisk oppbygning, menyer, fonter, skalering og gruppering av elementer.

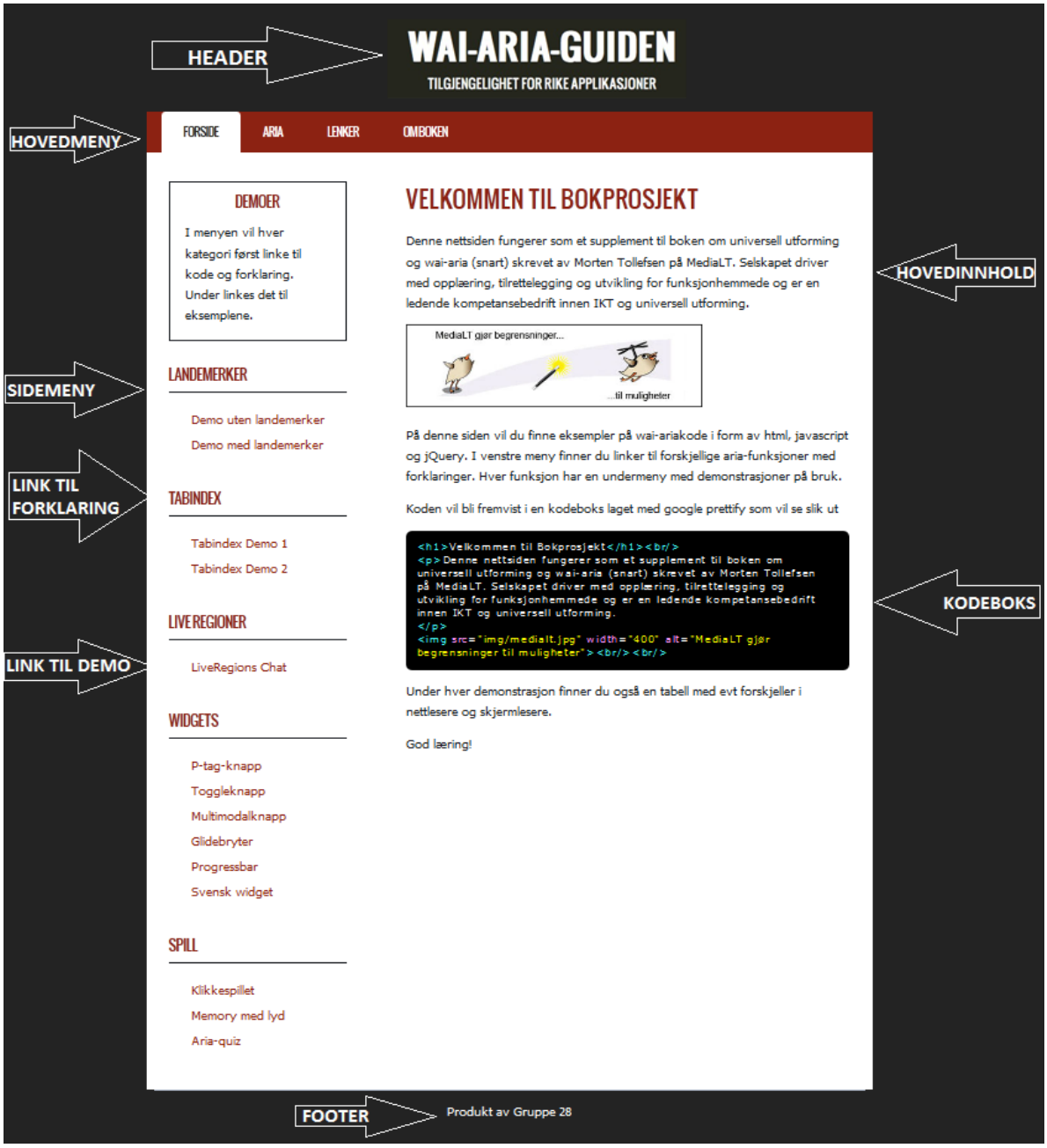
Nettsiden var i utgangspunktet ment for å supplere en bok om universell utforming, og det blir av den grunn nødvendig å endre design og menyer i henhold til hva som skrives i boken. Av denne grunn har hovedmenyens innhold ikke blitt særlig vektlagt og presenterer kun veldig generell informasjon om ARIA, prosjektet og noen lenker. Vi regner med at dette vil bli endret ved utvidelse av nettsiden.

Ved bruk av siden må man ta hensyn til at enkelte skjermlesere og nettlesere kan være innstilt på forskjellige måter og derfor gi et annerledes resultat enn forventet.

3. Nettsidens oppbygging

3.1 Struktur og innhold

Nettsiden er bygget opp på en tradisjonell måte med en header og logo, hovedmeny, hovedinnhold, sidemeny og bunntekst.



Logo linker til forsiden.

Hovedmeny linker til informasjonsbaserte sider:

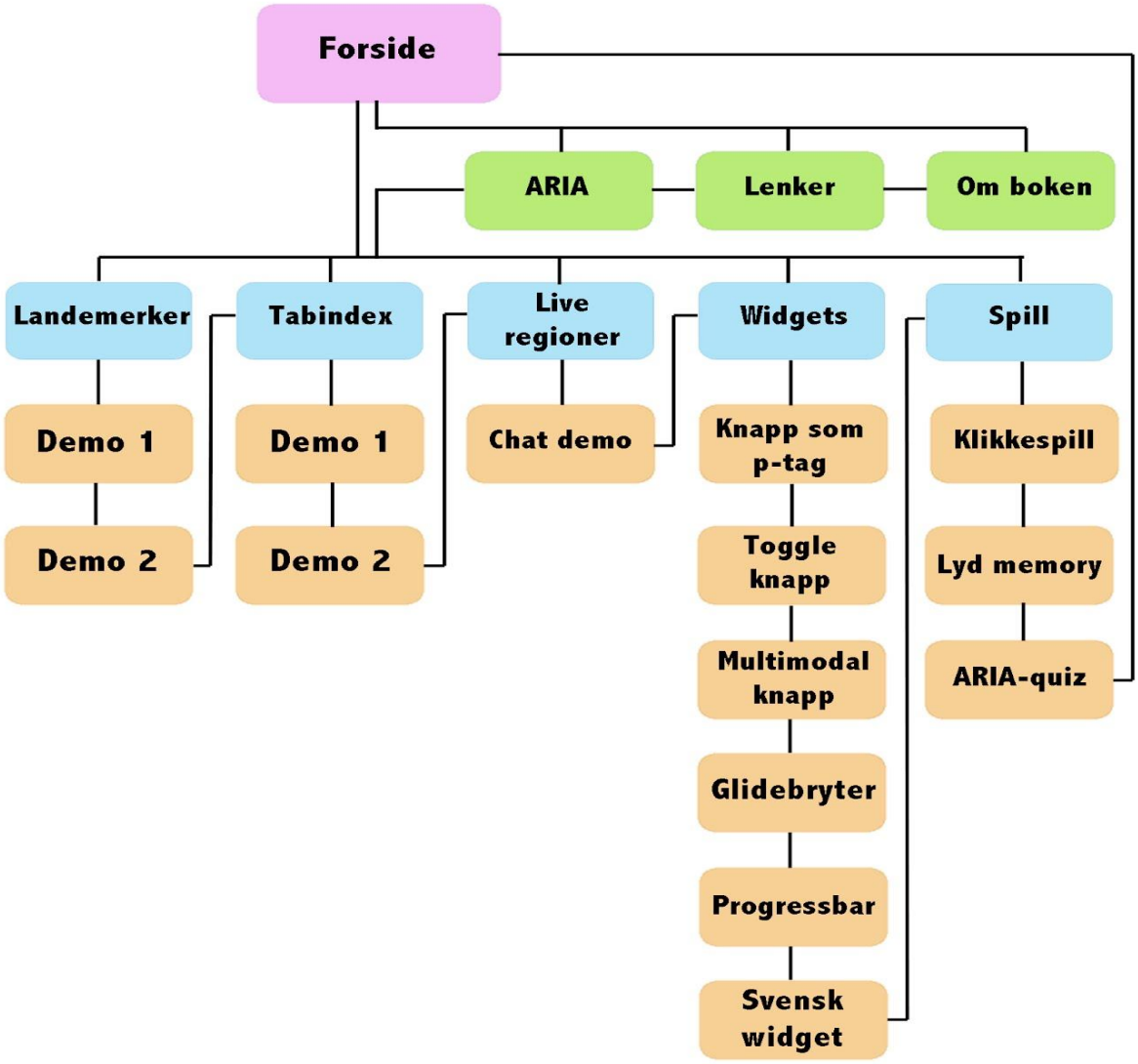
- **Forside** som forklarer litt om selve nettsiden.
- **ARIA** som beskriver WAI-ARIA.
- **Linker** viser relevante sider tilknyttet prosjektet.
- **Om boken** forklarer bakgrunnen for prosjektet.

Hovedinnhold er hvor all brødtekst(innholdet) på hver side ligger. Hovedinnhold bruker h1-tagger som overskrift til innhold og h2-tagger som underoverskrifter. Kode som blir presentert i hovedinnhold legges i kodebokser med svart bakgrunn og fargekoder på koden.

Sidemenyen består av store linker med informasjon om hvert ARIA-område og med undermenyer for de tilhørende demoene.

Footer linker til gruppenettsiden til Gruppe 28.

Domenemodell for nettsiden:


Forside
ARIA
Lenker
Om boken
Landemerker
Tabindex
Live regioner
Widgets
Spill
Demo 1
Demo 2
Demo 1
Demo 2
Chat demo
Knapp som p-tag
Toggle knapp
Multimodal knapp
Glidebryter
Progressbar
Svensk widget
Klikkespill
Lyd memory
ARIA-quiz

3.2 Design

3.2.1 WCAG

Nettsiden er utformet i henhold til WCAG 2.0 og er validert på WebAIM sine sider. WCAG kategoriseres i A, AA og AAA etter hvor strenge krav siden oppfyller. Det er 12 grunnleggende retningslinjer som skal oppfylles, med varierende delkrav under hver av disse:

1. Tekstalternativer: Gi tekstalternativer til alt ikke-tekstlig innhold, slik at det kan konverteres til formater som brukerne har behov for, for eksempel stor skrift, blindeskrift, tale, symboler eller enklere språk.
2. Tidsbaserte medier: Gi alternativer til tidsbaserte medier.
3. Mulig å tilpasse: Lag innhold som kan presenteres på forskjellige måter (for eksempel med enklere layout) uten at informasjon eller struktur går tapt.
4. Mulig å skille fra hverandre: Gjør det enklere for brukerne å se og høre innhold, blant annet ved å skille forgrunnen fra bakgrunnen.
5. Tilgjengelig med tastatur: Gjør all funksjonaliteten tilgjengelig med tastatur.
6. Nok tid: Gi brukerne nok tid til å lese og bruke innhold.
7. Anfall:Ikke utform innhold på en måte som er kjent for å forårsake (epileptiske) anfall.
8. Navigerbar: Gjør det mulig for brukerne å navigere, finne innhold og vite hvor de befinner seg.
9. Leselig: Gjør innholdet leselig og forståelig.
10. Forutsigbar: Sørg for at websider presenteres og fungerer på forutsigbare måter.
11. Inndatahjelp: Hjelp brukere med å unngå feil og å rette opp feil.
12. Kompatibel: Sørg for best mulig kompatibilitet med aktuelle og fremtidige brukeragenter, inkludert kompenserende teknologi.

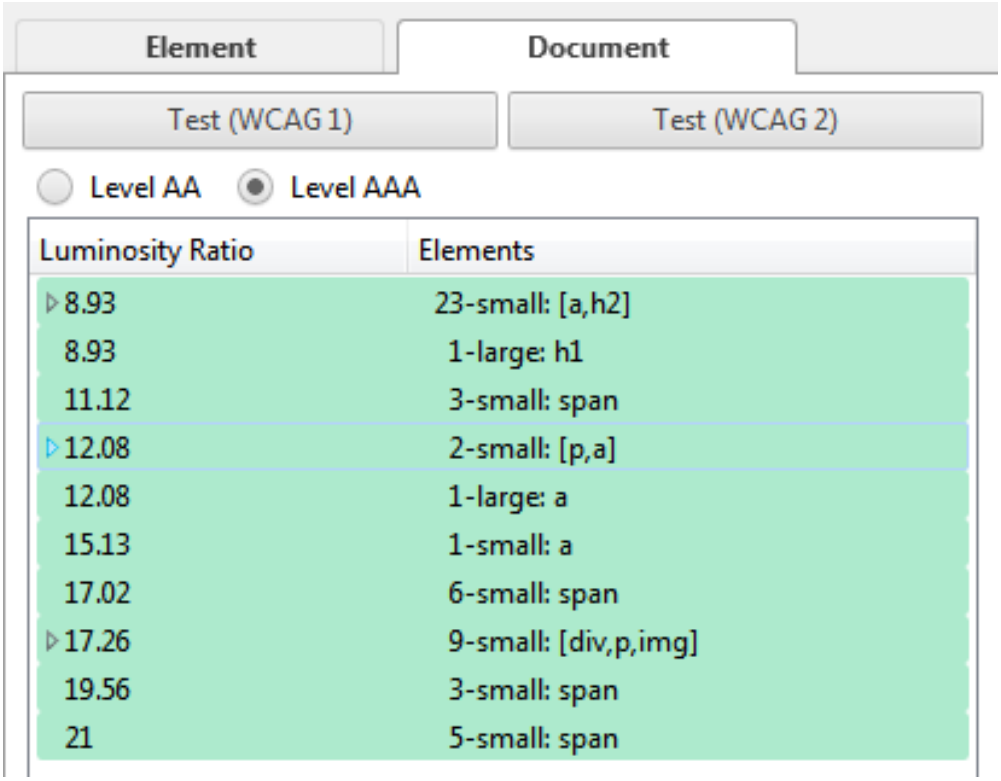
3.2.2 Tilpasninger til WCAG

Menyene er konsistente i hele nettsiden, unntatt demonstrasjonene hvor de er fjernet for og ikke å fremstå forstyrrende på tekstområdet.

Ved **kontrast** på nettsiden har vi forholdt oss til disse WCAG-kravene:

- Text and images of text have a contrast ratio of at least 7:1.
- Large text (over 18 point or 14 point bold) has a contrast ratio of at least 4.5:1

Alle elementene på nettsiden har disse kontrastforholdene ("Luminosity Ratio"):



Vi holder oss konsekvent på et kontrastforhold på minst 8.93:1. Dette er enda litt bedre enn kravet til WCAG level AAA om et kontrastforhold på minst 7:1.

Tekst som strekkes for langt utover skjermen kan være tung å lese, derfor er det normalt å gruppere tekst i rammer og kolonner som følger en logisk visuell struktur. Ettersom assisterende teknologi tolker nettsider hierarkisk ut i fra HTML-kode, kan dette føre til at teksten blir snudd på hodet. Vi har derfor valgt og ikke å sidestille hovedtekst. CSS brukes kun til å konsentrere det lesbare området til midten av siden. Font er sans-seriff som er enklere å lese.

Google-Code-Prettify

For presentasjon av hver kode har vi brukt **Google-Code-Prettify**. Dette kalles også beautifiers eller syntax highlighters og gir en bedre visuell fremstilling av kode. Kodeboksene er tilpasset svaksynte, og gir i tillegg logisk struktur ved hjelp av farge:

```
<img src="btn1.jpg"onclick="alert('Nå har du aktivert knappen med mus!');"/>
```

Fargekoder er det forøvrig ikke hensiktsmessig å formidle til en skjermleser da dette vil gi dobbelt så mye output.

For å kunne bruke Google-Code-Prettify på nettsiden kan skriptet hentes opp på denne måten:

<script src=”http://google-code-prettify.googlecode.com/svn/trunk/src/prettify.js” type=”text/javascript”></script>

I tillegg må vi kjøre funksjonen prettyPrint() - det gjøres slik:

<body onload="prettyPrint()">

Denne funksjonen vil nå modifisere utseendet på all tekst inne i elementet <pre class="prettyprint">. CSS brukes for å bestemme hvilke farger som skal benyttes. Dette har vi definert i vår standard CSS-fil med våre egne WCAG-vennlige fargekoder. Her er et eksempel på hvordan dette fungerer:

pre .str, code .str { color: #ffff00; } */ Gul farge */
pre .com, code .com { color: #AEAEAE; font-style: italic; } /* comment - grå */

Dette betyr at alt som er strings i koden vil få fargen gul og kommentarer vil bli grå.

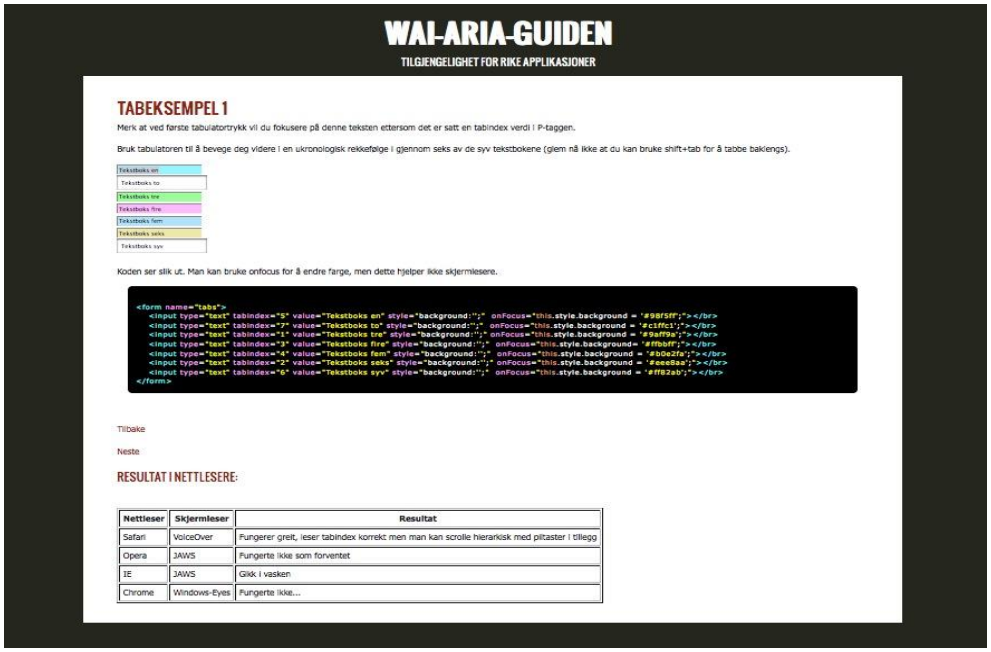
Språket på nettsiden har vi endret til en mer pedagogisk og lærerik tone og bevisst prøvd å unnlate overkompliserte setninger, fagutrykk og fremmedord som gjør teksten

mindre leselig. Skjermlesere kan nemlig få problemer med ord som ikke eksisterer i talesyntesen. Slik blir innholdet også enklere og mer intuitivt, noe som bedrer læreprosessen og inkluderer brukere med lettere kognitive utfordringer. Sidene er også merket med *<html lang="no">* for å unngå at skjermlesere velger feil språk slik at man ender opp med å få lest ut norsk med feks engelsk uttale.

På lengre avsnitt har vi hatt med noen enkle **illustrasjoner** i photoshop som et supplement til teksten, og også gruppert teksten i mindre deler med egne overskrifter. Det er viktig å kunne begrense seg slik at man ikke overkompliserer informasjonen og glemmer at man egentlig skriver for brukeren.

For god **navigering** er resten av nettsiden er merket med ARIA-landemerker for å definere hvilken funksjon hver tekstgruppe har. Disse landemerkene hjelper skjermleseren å tolke innholdet på siden slik at brukeren lettere kan orientere seg.

Demosidene ser litt annerledes ut i forhold til resten av nettsiden. På demosidene er hovedmenyen, sidemenyen og footer fjernet for å ha minst mulig forstyrrende elementer. Det som står igjen er header med logo som linker til forside, hovedinnhold med demo, og to avsluttende linker, en til forsiden og en til neste demo.



Hver demoside har forklaring på kode, utenom enkelte hvor man allerede finner forklaring i hovedmenyen. Man får også presentert den aktuelle kode, og en

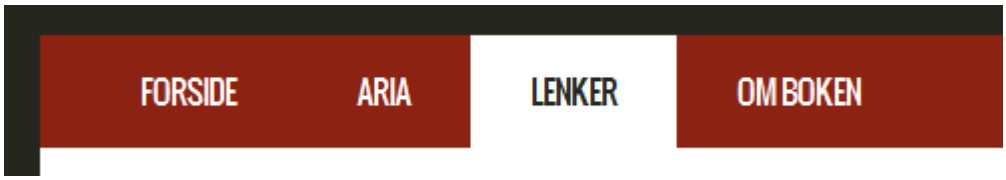
tabelloversikt over testresultater i kombinasjon med forskjellige nett- og skjermlesere. Dette er nødvendig for å påvise sprikende oppførsel.

3.3 Bruk av PHP

PHP blir brukt for å bruke maler på en enkel måte.

Maloppbygningen består av top.php og bottom.php. Disse inneholder koden som kreves for å bygge opp det som er markert på bildet over. I tillegg inneholder top.php metainformasjon som står i head-elementet, som tittel på siden, css-filer og så videre.

Variabelen $current bestemmer hvilket toppmenyelement som skal ha en hvit bakgrunnsfarge - som indikerer hvilken side som er aktiv. Hvis current er satt til lenker vil det se slik ut:



Med en kort if-setning sjekkes det hva som er $current, og CSS-klassen currentMenu settes for å endre utseendet.

<li tabindex="3"<?php if ($current == 'lenker') { echo ' class="currentMenu"'; } ?>>Lenker

En lignende metode brukes for å sette tittelen på siden:

<title><?php echo $title; ?></title>

4. Beskrivelse av demoer

Under hver kategori i nettsiden har vi demonstrasjoner på god ARIA-praksis. Her oppsummerer vi hva disse testområdene dreier seg om og hvordan de er blitt utført.

4.1 Landemerker

Disse demonstrasjonene viser effekten av landemerker for å sette holdepunkter i en side. Slik kan man gruppere nettsider inn i logiske felt som ellers kun er synlige ved hjelp av CSS, slik som banner, menyer o.l.

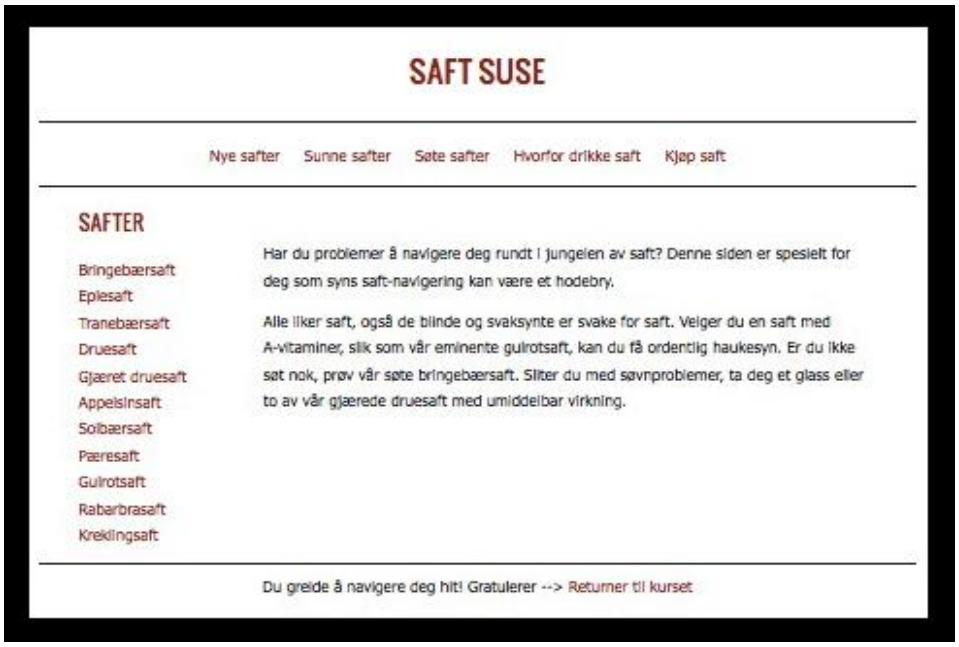


Fig x. En fiktiv nettside som testes med skjermleser

Vi har laget to separate demoer med og uten landemerker ettersom det ikke er mulig å demonstrere forskjellen på samme side da en nettside tolkes helhetlig. For å få riktig forståelse av dette er det nødvendig å teste disse demoene ved hjelp av skjermleser, ellers vil de oppleves som identiske.

4.2 Tabindex

Tabindex er et hjelpemiddel som lar utvikleren kontrollere bruken av fokus via JavaScript og tabulator-navigering enten sekvensielt eller sette egenspesifisert rekkefølge på elementene. Men det skal sies at man bør være forsiktig med å bruke tabindex ettersom man kan ødelegge flyten og den logiske strukturen i hvordan tabulatoren beveger seg og hemme brukerens opplevelse, særlig ved å gjøre elementer ufokuserbare. Tabindex er ellers et godt verktøy om det brukes korrekt. Utfordringen er å vite hvilke elementer som allerede mottar fokus fra nettleseren, og hvilke som allerede mottar fokus fra skjermlesere.

Her er en forklaring på de forskjellige tabindex-verdiene:

Tabindex-verdi	**Hendelse**
Uten tabindex-attributter i koden	**Tabulator beveger seg sekvensielt gjennom koden.** Tabulatoren vil bevege seg sekvensielt gjennom de standard autofokuserbare elementene (linker, knapper og input-felt) fra topp til bunn etter nettsidens kode.
tabindex=”0”	**Tabulator beveger seg sekvensielt gjennom koden og gjør ikke-fokuserbare elementer fokuserbare.** Tabulatoren vil bevege seg sekvensielt gjennom de standard fokuserbare elementene (linker, knapper og input-felt) fra topp til bunn etter nettsidens kode, ettersom verdi “0” ikke endrer mønsteret tabulatoren beveger seg i. Ved bruk av skjermlesere kan viktig innhold i uviktig innhold lett hoppes over, men ved å sette en index-verdi vil dette sette et fokus på det viktige innholdet. Du kan og gjøre elementer som normalt ikke er fokuserbare til å bli det ved å sette inn en tabindex-verdi, for eksempel i en p-tag Hello World .

tabindex=”1 - 32767”	**I rekkefølge før 0 og uten tabindex-atributt.** Ved å sette inn en verdi fra “1” - “32767” vil dette føre til at tabulatoren beveger seg gjennom disse før alt annet, rangert fra det laveste til det høyeste nummeret uansett hvor elementene ligger i koden. Når tabulatoren har beveget seg gjennom disse vil den fortsette å bevege seg sekvensielt gjennom de tabulerbare elementene uten tabindex-attributter og de med en verdi på “0”. Ved å sette sammen kun en verdi på flere elementer, for eksempel sette tabindex=”1” på tre forskjellige linker, vil dette føre til at tabulatoren beveger seg sekvensielt i koden gjennom disse før den beveger seg videre til et høyere index-tall osv.
tabindex=”-1”	**Gjør fokuserbare elementer ufokuserbare og setter focus gjennom JavaScript.** Ved å sette inn verdien “-1” på et fokuserbart element vil dette føre til at elementet blir ufokuserbart, med andre ord vil du ikke ha mulighet til å kunne bruke tabulatortast til elementet. Den fungerer og på en annen måte når det kommer til JavaScript. “Div’er” i JavaScript er normalt ikke fokuserbare men kan gjøres fokuserbare ved å sette tabindex=‘-1’, ved å gjøre dette kan man bruke .focus-funksjonen og når da scripet kjøres vil “div’en” bli satt i fokus. Et eksempel kan være at man trykker på en svar-knapp på en quiz og resultatet vil poppe opp med fokus rundt.

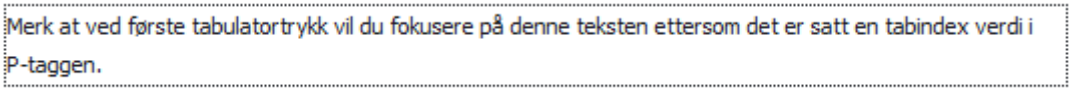
4.2.1 Tabindex demoer

For å gi en beskrivelse av tabindex sine funksjoner har vi laget to enkle demoer.
Tabindex satt for å gi fokus gjennom JavaScript er brukt i “Aria-Quiz” og er forklart i forklaringen til den (se punkt “4.5.3”).

Demo 1

I det første eksempelet vil gruppen illustrere hvordan man kan sette fokus på en p-tag og hvordan man kan endre den sekvensielle rekkefølgen til tabulatoren. Eksempelet utføres ved å tabbe 8 ganger fra du kommer inn på nettsiden. Eksempelet er kun kodet i HTML og CSS.

Eksempelet viser først en p-tag (tekstfelt) som får fokus ved første tab-trykk, p-taggen er satt til tabindex=”1”.



Screenshot av Tabeksempel 1, bilde av tekstfelt med fokus.

Videre bruker man tabulator gjennom 7 tekstbokser i en usekvensiell rekkefølge hvor fargene endrer seg når man passerer hver av boksene. Hver av tekstboksene har fått en unik tabindex-verdi fra “1”-”7”.

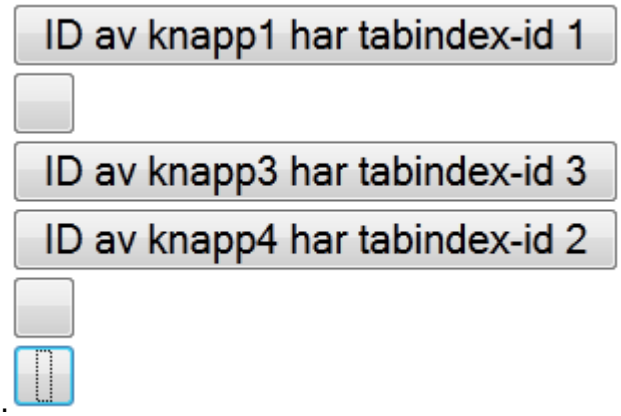
Screenshot av Tabeksempel 1, bilde av tekstboksene som tabbes igjennom.

Demo 2

I eksempel 2 illusterer gruppen igjen hvordan man kan bruk tabindex til å endre den sekvensielle rekkefølgen. Demo 2 er kodet i HTML og JavaScript.

I eksempelet skal man tabbe seg igjennom 6 knapper, knappene gir tilbakemelding om hvilket tabindex-nummer de har om man trykker på dem. Man vil komme frem til at det

er en knapp det **ikke** går an å manøvrere seg til på denne måten, klikker man på den med en mus ser man hvorfor, den er satt med tabindex=”-1”

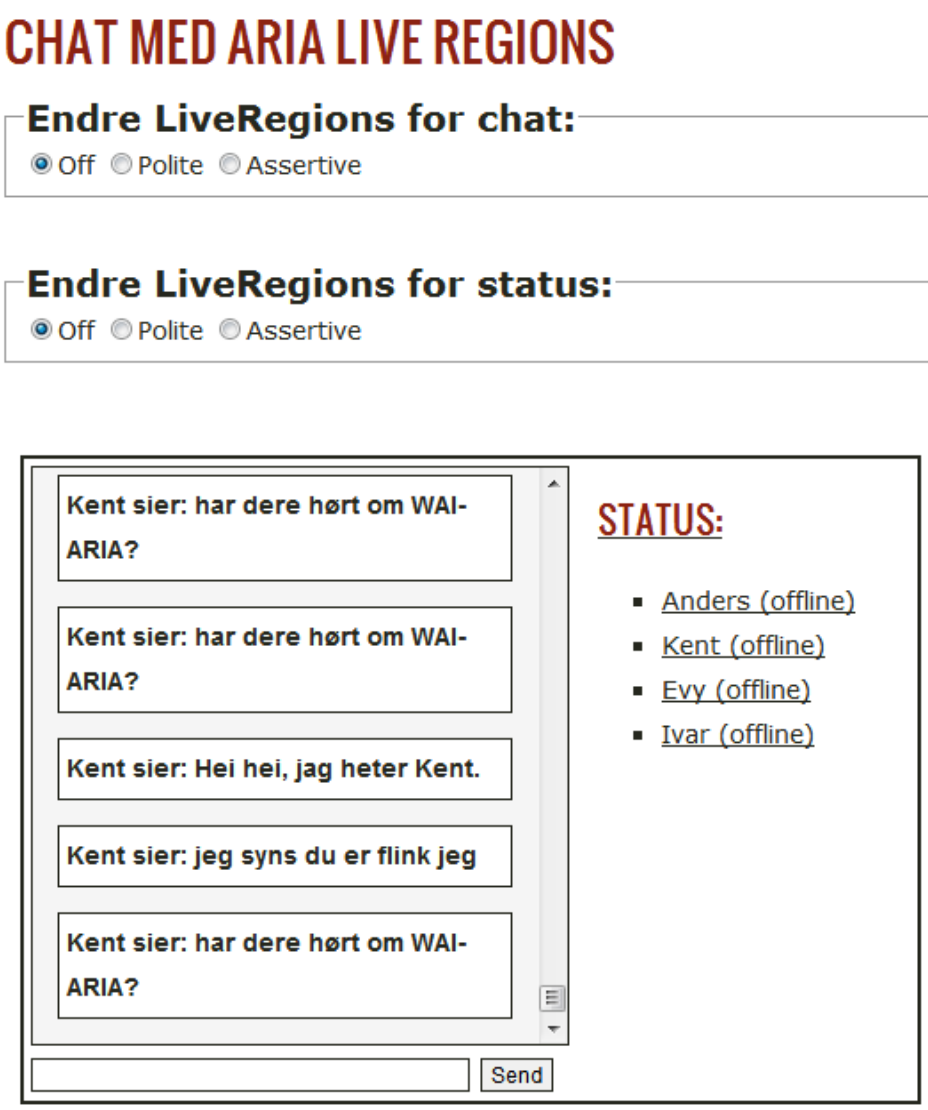


Screenshot av Tabeksempel 2, bilde av knapper som tabbes igjennom og viser tabindex-verdi.

4.3 Liveregioner

For å gi et beskrivende eksempel på liveregioner har vi utviklet en fiktiv chat-applikasjon.
Denne demonstrasjonen viser bruk av liveregioner i en applikasjon som sender ut oppdateringer til tilfeldige tider. Målet med demoen er å vise hvordan en skjermleser oppfatter disse endringene med forskjellige liveregion-innstillinger.

4.3.1 Chatapplikasjon



En chat byr på mange utfordringer for brukere av skjermlesere. Problemet ligger først og fremst i å vite hva man skal få opplest, når og hvor ofte. WAI-ARIA bruker liveregion-attributten for å fortelle skjermleseren i hvilken grad den skal lese opp nye statusoppdateringer og chatmeldinger. Løsningen blir å sende ut tekst til elementer som har aria-live aktivert med en politeness-instilling ("høflighetsgraden").

For best mulig opplevelse anbefales det at demoen testes med en skjermleser som støtter WAI-ARIA, f.eks. NonVisual Desktop Access (NVDA) eller JAWS.

Brukeren kan her velge mellom flere forskjellige livregion-attributter gjennom radioknapper:

Liveregionene er opprinnelig satt til off. De forskjellige valgene vil ha følgende virkning:

- **"off"** deaktiverer Live-Regioner.
- **"polite"** er høflig og aktiveres i skjermleseren bare når det ikke er noe annet som blir opplest.
- **"assertive"** aktiveres med det samme nytt innhold dukker opp.

```
$('.endre_chat').change(function(){
   var valg = $('input[class=endre_chat]:checked').val();
   $('#prat').attr('aria-live', valg);
   alert('Live-regionen til pratefeltet er endret til ' + valg)
});
```

De to feltene "prat" og "brukerliste" oppdateres etter en intervall på henholdsvis 3 og 5 sekunder, fastsatt i JavaScript.

```
setInterval(randomStatusOppdateringer, 5000);
setInterval(randomTekst, 2000);
```

Det er disse feltene i bildet over som får sin liveregion-attributt satt gjennom radioknappene slik at skjermlesere leser opp oppdateringene.

Knappen "Send" har noen ekstra attributter som gjør applikasjonen mer tilgjengelig:

- role="button"
- aria-controls="prat"

Rollen button gjør det helt klart at knappen faktisk brukes som en tradisjonell knapp. Aria-controls forteller nettleseren at trykk på "Send"-knappen vil føre til endringer i pratefeltet - f.eks. "Du sier: Hei".

Prate- og statusfeltet er også merket med *aria-atomic="false"* slik at bare nye elementer blir lest opp. Dette er egentlig standardverdien, men vi ønsker å klargjøre dette ved å sette verdien manuelt.
Pratefeltet har i tillegg role-attributten satt til *"log"*. Dette beskriver et felt hvor ny informasjon kommer opp og gammel informasjon kan forsvinne. Det kreves at den nye informasjonen kommer i en meningsfylt rekkefølge, f.eks. kronologisk, som ofte er tilfellet for chat-felt.

4.4 Widgets

4.4.1 Knapp som p-tag

I denne demonstrasjonen er poenget å vise hvordan man kan lage en widget ut av noe som vanligvis ikke er fokuserbart. I dette tilfellet lager vi en knappefunksjon ut av noe som i utgangspunktet ikke er en knapp og heller ikke mottar fokus. Ved å sette *role="button"* gir dette en ARIA-verdi som forteller assisterende teknologi at dette er en knapp. Ellers er knappen en img src-tag. Man kan fortsatt ikke bruke tabulator til knappen, men ved å sette *tabindex="1"* vil den også kunne bli markert. Ved JavaScript kan man oppnå samme effekt som en knapp ved å sette en OnKeyDown event.

4.4.2 Toggle- og multimodale knapper

De påfølgende to knappedemoene viser eksempler på aria-atributter man kan benytte på toggle- og multimodale knapper for å fortelle om status som ellers kun gis ved hjelp av farge. Disse kan gis en rolle som checkbox og forskjellige verdier som *ARIA-checked="true"/"false"/"mixed"*.

Fig x. Multimodale knapper med ARIA-verdier og forskjellig uttrykk

Man kan også inkludere ARIA i CSS ved å bruke klammeparenteser rundt egenskapene for å endre utseende.

4.4.3. Svensk widget

Vi lagde også en liten widget som heter svensk efternamn-generator som tar i mot input og presenterer et resultat gjennom JavaScript ved å koble sammen to array-verdier med

en random-funksjon. Poenget her er å demonstrere at det som virker logisk visuelt ikke alltid fremstår like logisk for en skjermleser. Dette må derfor tilpasses slik at alle kan få samme utbytte. I denne demonstrasjonen har man en knapp som aktiverer et input-felt (tar imot fornavnet ditt). Når scriptet har kjørt vil output-feltet (ditt nye svenske navn) dukke opp på samme sted som knappen var og dermed forflytte knappen ned, noe som visuelt gir mening ettersom det følger leseretningen.

Dette vil forøvrig ikke være den beste løsningen for en skjermleser, som stadig har markøren på knappen og ikke får med seg teksten vist ovenfor. Slike problemer kan man løse ved hjelp av focus() funksjonen i JavaScript for å flytte markøren til korrekt sted.

4.4.4. Glidebryter

Glidebrytere er nytt i HTML5 og kan nå benyttes uten hjelp av JavaScript. Vi har testet denne funksjonen sammen med ARIA.

4.4.5. Framdriftslinje

Vi har også utforsket framdriftslinjen og hvordan denne best kan formidle status utover det visuelle. “Progress” er et nytt element i HTML5. Elementet brukes til å vise framdriften til en oppgave. At et slikt element er en del av standarden, gjør det er mye enklere å utføre programmeringen/skriptingen som trengs for ha en indikator på

progresjonen av en oppgave. Her demonstreres hvordan ARIA-attributter kan brukes for å gi en mer hensiktsmessig presentasjon av framdriften enn det en vanlig framdriftslinje gjør.
Standardpresentasjonen av "progress" i visuelle nettlesere er en framdriftslinje:

For at progresjonen ikke skal se hakkete ut, bør linjen oppdateres minst 20 ganger i sekundet (dvs. hvert 50. millisekund). En slik oppdateringshastighet er selvsagt altfor høy for skjermlesere, derfor må verdiene som brukes av skjermlesere oppdateres med et annet intervall. Vi har forsøkt å gjøre dette ved å bruke ARIA, med blandet resultat. IE og Safari støtter ikke "progress"-elementet, og i Opera virker ikke skjermleseren NVDA. Resultatene her er, inntil annet nevnes, oppnådd med Firefox og NVDA.

I henhold til HTML5 har en del html-elementer en standard eller en påkrevet aria-rolle[1]. Elementet "progress" har påkrevet aria-rolle "progressbar", så det skulle være unødvendig å eksplisitt gi elementet denne rollen. Faktisk er det ikke engang tillatt; W3Cs HTML5-validator sier tydelig at "progress" ikke kan tildeles noen ARIA-rolle. Problemet er at når rollen "progressbar" ikke er satt, tar skjermleseren NVDA tydeligvis sine verdier fra "progress"-elementets "max"- og "value"-attributter, og forsøker å beregne hvor stor andel av oppgaven som er fullført. Den beregningen vil i de fleste tilfeller gi et stort antall desimaler, noe som igjen får NVDA til å gå i spinn og lese opp feilaktige og meningsløse verdier. Når rollen "progressbar" settes, og attributtene "aria-valuemin", "aria-valuemax" og "aria-valuenow" brukes til å gi NVDA ARIA-verdier å forholde seg til, forsvinner dette problemet. For at siden skal validere som HTML5, har vi "jukset" ved å sette ARIA-rollen og de øvrige ARIA-attributtene i et skript (Siden er likevel egentlig ugyldig HTML5, men dette knepet lurer validatoren).

Dette eksempelet har også andre problemer. Vi har satt "aria-live" til "assertive", med ønske om at skjermleseren skal lese opp verdien hver gang en endring inntreffer (dvs. hvert annet sekund). Det skjer ikke med NVDA; for å få lest opp verdien må musepekeren trekkes over framdriftsindikatoren hver gang en verdi skal leses opp. For å kunne fokusere framdriftsindikatoren med tab-knappen eller museklikk, har vi gitt

1 http://www.w3.org/TR/html5/wai-aria.html#wai-aria

den “tabindex”-verdien 0. Av en eller annen grunn fører det til at den fokuseres to ganger i løpet av tab-syklusen. Uten tabindex=0 lar den seg ikke fokusere med tab-knappen (eller museklikk) i det hele tatt. Det er også noen påfallende forskjeller mellom hva som leses opp når musepekeren føres over elementet, og hva som leses opp når elementet får fokus med tab-knappen eller museklikk.

Resultater i andre nettlesere:
I Safari, med Macs innebygde skjermleser VoiceOver, oppsto den interessante effekten at selv om <progress> ikke støttes av Safari (5.1), klarte skjermleseren å lese opp progresjonen som prosent fullført, selv om statusen er usynlig. Noe tilsvarende er ikke observert med noen annen kombinasjon av nettleser og skjermleser.

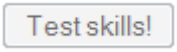
4.5 Blandede applikasjoner

Gruppen har laget en liten seksjon med litt spill og moro, hvor du kan leke og hygge deg. Alle spillene er laget ARIA-vennlige, men kan kanskje fremstå litt forskjellig avhengig av type nettleser ettersom hvilke støtte de har for den typen ARIA. Klikkespillet tester hvor mange ganger du greier å taste kontinuerlig på tastaturet i 10 sekunder. Du kan også prøve Memory med lyd og gjenkjenne lyder. Har du tilgang til skjermleser kan du gjøre skjermen mørk og prøve orientere deg fra lyd til lyd. Til slutt er det laget en WAI-ARIA-quiz for å teste kunnskapene man har tilegnet seg på nettsiden om WAI-ARIA.

4.5.2 Klikkespill

Klikkespillet er et veldig enkelt spill, alt det går ut på er å trykke så mange ganger man kan på tastaturet i 10 sekunder og for hvert trykk får man 1 poeng.

Spillet starter ved at man trykker på knappen 

, denne har autofokus når man kommer inn på siden så det er bare å trykke ENTER for å aktivere. Dette starter en nedteller i bakgrunnen av nettsiden som starter å tikke i 10 sekunder i tillegg til at den aktiverer knappen 

som man trykker på for å få poeng. Videre trykker man enten TAB eller markerer "Test skills!"-knappen med mus for å få fokus på knappen. Det anbefales å trykke TAB siden man ikke får poeng av å trykke på knappen med mus. Når knappen er markert er det bare å slå løs på alle taster som ikke har en snarvei eller funksjon som aktiveres. Dette kan forøvrig være et problem når man bruker skjermleser ettersom de enten overkjører tastaturet med nye funksjoner eller deaktiverer knappen.

Klikkespillet er laget i JavaScript og HTML. Ettersom dette er en dynamisk nettside hvor innhold dukker opp uten at siden oppdaterer seg er det lagt til LiveRegions for å gi tilbakemelding til skjermleseren. Dette er lagt til i score og i gratulasjonstekst.

Spillet ble laget tidlig i prosjektet i sammenheng med selvopplæring i JavaScript og for å teste live-regionens funksjonalitet med en skjermleser.

4.5.2 Memory-spill

Spillet er modellert etter det klassiske huske-spillet der det gjelder å finne par med like motiver. Slike spill er vanligvis lagt opp til at en spiller snur ett kort, og deretter skal snu et annet kort som skal være likt det første. Før start stokkes kortene og legges på bordet med baksiden opp. Hvor parene befinner seg ved spillets begynnelse er dermed helt uvisst.
Spilleren får ved hvert forsøk muligheten til å snu to kort. Hvis spilleren klarer å snu to like kort, innkasseres de to kortene, og spilleren får prøve å finne et nytt par. Hvis spilleren feiler, snus kortene tilbake og turen går videre til neste spiller. I begynnelsen beror muligheten til å finne to like kort utelukkende på flaks. Etterhvert som kortene blir snudd kommer hukommelseselementet sterkere inn, siden spillerne får se hvor en del av kortene befinner seg. Spillerne som best husker hvor de forskjellige kortene er, får et klart fortrinn, og vil med stor sannsynlighet vinne spillet (Vinneren er den som

har klart å samle inn flest par under spillets gang). Et slikt spill kan ha et ubegrenset antall spillere, eller man kan spille det alene. I det siste tilfellet blir utfordringen bare å klare seg med så få feil som mulig.

Spillet i dette eksempelet bruker lyd i stedet for bilder, så her er det om å gjøre å huske lyder. Videre er det laget for å spilles av kun én spiller. Et viktig mål har vært at spillet i så stor grad som mulig skal være nøytralt i forhold til spillernes synsevne. På bakgrunn av dette har vi forandret litt på spillet fra hvordan det er beskrevet ovenfor.

De vesentligste egenskapene er som følger:

- Kortene snus ikke. I stedet spilles en lyd når spilleren klikker på kortet (Å "klikke på" betyr her å enten klikke på det med musepekeren, eller å trykke enter eller mellomrom når kortet har fått fokus ved bruk av (skift-)tab.)
- Lyden spilles av en gang ved klikking og kortet kan klikkes på så mange ganger man ønsker.
- Kriteriet for å finne et par er ikke bundet til at kortene snus/klikkes parvis, det er tilstrekkelig at kort med to like lyder klikkes rett etter hverandre.
- En "funnet-lyd" spilles av når to like lyder er klikket rett etter hverandre.
- Par som er funnet, blir ikke fjernet. De markeres i stedet visuelt og auditivt; klikk på slike kort gir en egen “boing-lyd” som indikerer at de allerede er funnet, og markeres i tillegg med en stjerne. På denne måten oppnås tilgjengelighet for alle brukere.
- Når alle parene er funnet, spilles det av en “klappe-lyd” som markerer at spillet er ferdig.
- Et nytt spill, med en ny sammensetning av lyder, kan alltid startes ved å trykke "Nytt spill"-knappen (eller oppdatere siden).

Spillets oppbygning og tilgjengelighet

Knappene som spillet består av, er plassert i en tabell. Til hver knapp er det lagt et bilde som er en enkel grafisk illustrasjon av et kort som ikke er snudd/funnet. Disse bildene har "alt"-attributtet "Ikke funnet". "alt" er et standard-attributt for bilder i HTML, og viser seg i de tilfellene der bildet ikke kan vises, f.eks. i tekstbaserte nettlesere, og blir også lest opp av skjermlesere.

Knappene har hver sin unike identifikator, som brukes av skriptet som startes av knappetrykk. Knappene kan være HTML-elementer av typene "input" med attributtet type="image", eller "button" med bildet lagt inn mellom start- og slutt-tagen. Å lage knappene som "img"-elementer fungerer ikke godt. Forskjellen mellom "input" og "button" i forhold til "img" er at sistnevnte ikke tar genererer onClick-hendelser ved bruk av tastatur, mens "input" og "button" behandler trykk på enter- og mellomrom-tastene som museklikk, og dermed utløser onClick-metodene. Dessuten er knappene logisk sett knapper/input, og ikke bilder. Vi har brukt "button"-elementer, men "input"-elementer gir så å si lik funksjonalitet.

Spillet har fått ARIA-rollen "application", som skal gi hjelpeteknologi et hint om gå inn i en modus tilpasset applikasjoner.
Tabeller lar seg normalt ikke navigere med piltaster uten bruk av skript laget for formålet. Vi har gitt tabellen, tabellradene og tabellcellene ARIA-rollene hhv. "grid", "row" og "gridcell" i håp om at ARIA-kompatible nettlesere ville gjøre tabellen navigerbar med piltastene. Det har vist seg å ikke fungere, men vi lar det bli stående i håp om at denne funksjonaliteten kommer etterhvert.

I de to nederste radene er knappene gitt ARIA-rollen "button" og ARIA-tilstanden aria-pressed="false". For å vise forskjellen som ARIA utgjør, er disse ARIA-attributtene sløyfet i de to øverste radene.

Skriptet som startes ved klikk på knappene, sjekker om knappen allerede er funnet som en del av et par, eller om klikket er det andre av to påfølgende på forskjellige knapper med samme lyd. I alle tilfeller spilles det av en lyd tilpasset den aktuelle (ev. nye) tilstanden til knappen og spillet. Hvis knappetrykket er det andre av to på forskjellige knapper med samme lyd, skal knappen (og den foregående) endre tilstand. Skriptet endrer da bildet til et som viser at knappene er en av et funnet par, samtidig som "alt"-teksten endres til "Funnet". For knappene som har ARIA-attributter, dvs. i de to nederste radene, endres "aria-pressed" til "true".

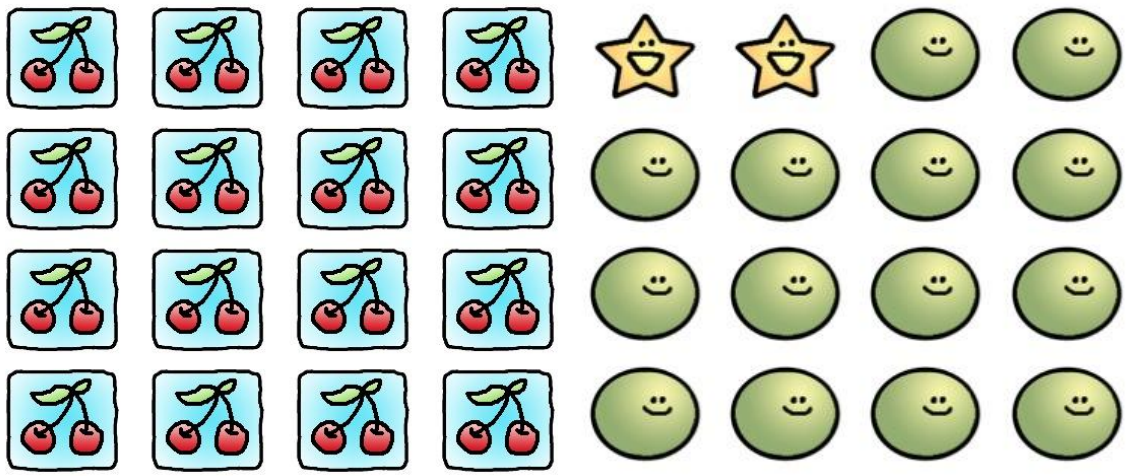
Testing av spillet

Ved vanlig, visuell bruk, fungerer det stort sett etter forventningene(*). Når skjermlesere brukes, og når navigeringen gjøres med tastatur er resultatet ikke helt tilfredsstillende.
Tastaturnavigering fungerer bare med tab-tasten forover i tabellen, bakover brukes skift-tab. Piltastene kan ikke brukes. Noe avhengig av nettleser og nettlesertillegg som brukes til lydavspilling, forsvinner fokus fra den klikkede knappen når lyden spilles av. Det er svært irriterende.

Skjermlesere forteller tydelig hvilken rad og kolonne som har fokus, men opplysninger om knappenes tilstand er i beste fall utydelig, i verste fall forvirrende og tildels villedende.

Noen nettlesere vil ikke spille av lydene hvis avspilleren settes til visibility:"hidden" i CSS, eller hvis “hidden”-attributtet settes til “true” i “embed”-/”object”-tagen. Derfor har vi måttet la den være et synlig og visuelt forstyrrende element.

Knappene ble først utformet med tilfeldige illustrasjoner, bl.a. kirsebær lik dem man ser på et casino. Ved å multiplisere det samme bildet 16 ganger fikk vi derimot en litt svimlende effekt, noe som kan forårsake epileptiske anfall hos enkelte, nesten inkludert oss selv. Vi besluttet å endre knappene til noe som var mildere for øynene.



Vi har beholdt en viss størrelse på de klikkbare flatene for større treffsikkerhet ved bruk av mus. Disse kan selvsagt skaleres slik man ønsker ved hjelp av nettleserens innebygde funksjoner. Lydene er lagret i MP3-filformat som er det vanligste lydformatet, for å redusere nødvendigheten for ekstra programvare.

4.5.3 Quiz

Quizapplikasjonen er et verktøy for å øke læreverdien til resten av nettsiden. Den består av 18 spørsmål om WAI-ARIA. Dataene er bygd opp på følgende måte:

```
var quiz =
  [
   {
    question: 'Hva er WAI-ARIA forkortelse for?',
    answers:
    [
      'WWW Accessibility Information - Advanced Rich Internet Accessibility', // 0
      'WWW Accessibility Information - Accessible Reader Internet Applications', // 1
      'Web Accessibility Initiative - Accessible Rich Internet Applications', // 2
      'Web Accessibility Initiative - Advanced Reader Interactive Applications' // 3
    ],
    correct: 2, // Correct peker til posisjonen det riktige svaret har i answers (første svar starter på 0)
    explanation: 'WAI står for <b>Web Accessibility Initiative</b> og ARIA står for <b>Accessible Rich Internet Applications</b>. WAI er en samling protokoller for tilgjengelighet på nett, og ARIA er deres tiltak for å øke tilgjengelighet rundt rike applikasjoner.'
   }
```

Selve skriptet kan støtte flere oppgavesett ved å deles opp i to deler:

- quiz.js - forblir alltid den samme

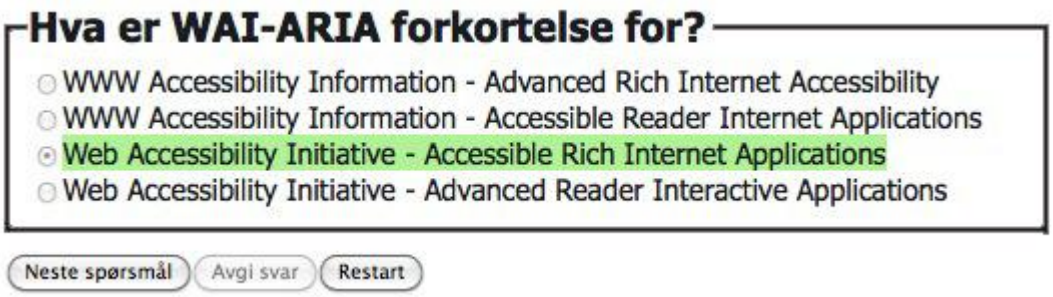
- <navnPåVOppgavesett>.js - inneholder et array med objekter som inneholder all den nødvendige informasjonen for hvert spørsmål

Programmet kan dermed settes sammen gjennom å referere til de i HTML-filen:

```
<script src="oppgavesett1.js"></script>
<script src="quiz.js"></script>
```

Det hadde vært mindre effektivt om alt hadde ligget i en fil fordi brukeren slipper å laste ned quiz.js hver gang. I tillegg blir også det fremidige arbeidet med å sette opp, oppdatere og organisere filene lettere.

En enkelt oppgave blir presentert på denne måten:



For å lage et tilgjengelig skjema som gir mening for alle brukere har vi fulgt WCAGs suksesskriterie 1.3.1 som sier følgende:

Info and Relationships: Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text.

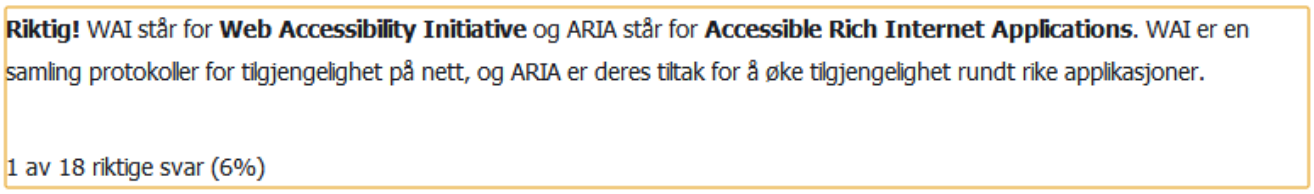
For å oppnå dette har vi tatt i bruk elementene “fieldset” “legend” og “label”. Et fieldset representerer en gruppe elementer som hører sammen. Legend er overskriften til grupperingen som i vårt tilfelle er spørsmålet.

En label gjør det mulig å koble en radioknapp opp mot en tekst, slik at det ikke bare er det visuelle som avgjør sammenhengen.. Slik fungerer det:

<input **id="alternativ00"** value=”0” type=”radio”>
<label **for="alternativ00"**>WWW Accessibility Information - Advanced Rich Internet Accessibility</label>

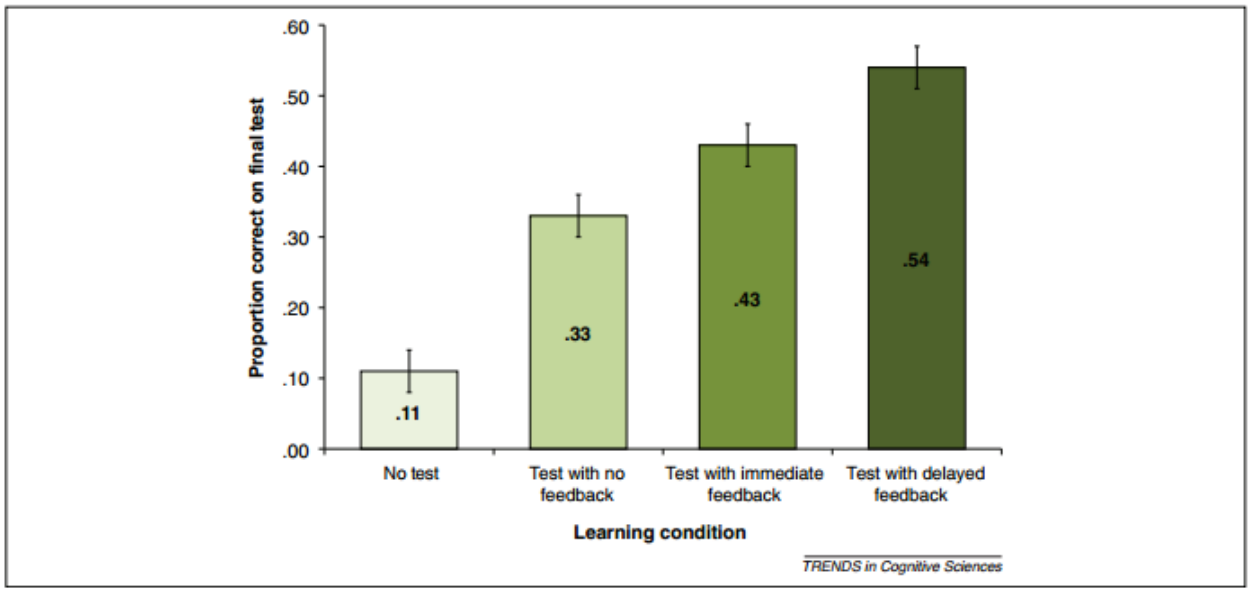
Label for=”alternativ00” refererer her selvfølgelig til radioknappens ID.

Når brukeren velger et svaralternativ ved å huke av en radioknapp må knappen “avgi svar” trykkes for å gå videre. Knappen kjører en sjekk som forhindrer muligheten for å svare blankt og deretter vurderer om svaret er riktig eller galt. Videre kommer det opp en melding med en forklaring på svaret, om det var riktig eller galt og antall foreløpig riktige svar.



Fokuset blir automatisk flyttet ned til denne ved å utføre resultatDiv.prop('tabindex', '-1') som gjør elementet fokuserbart med resultatDiv.focus();. Selv om feltet også er definert som en liveregion kan man bruke denne ekstra teknikken for å forsikre seg om at alle skjermlesere (også de som ikke støtter WAI-ARIA) faktisk leser opp det som står i elementet. Dette vil vært et problem i andre applikasjoner dersom man avhenger av å ha fokus et annet sted for å bruke applikasjonen.

Vi ønsket å gi brukeren nyttig og lærerik tilbakemelding underveis for å øke læreverdien. I en undersøkelse utført av professorer ved Memory Lab ved Washington University viser det seg at dette er den mest effektive måten å lære fra en quiz (Roediger & Butler, 2010).



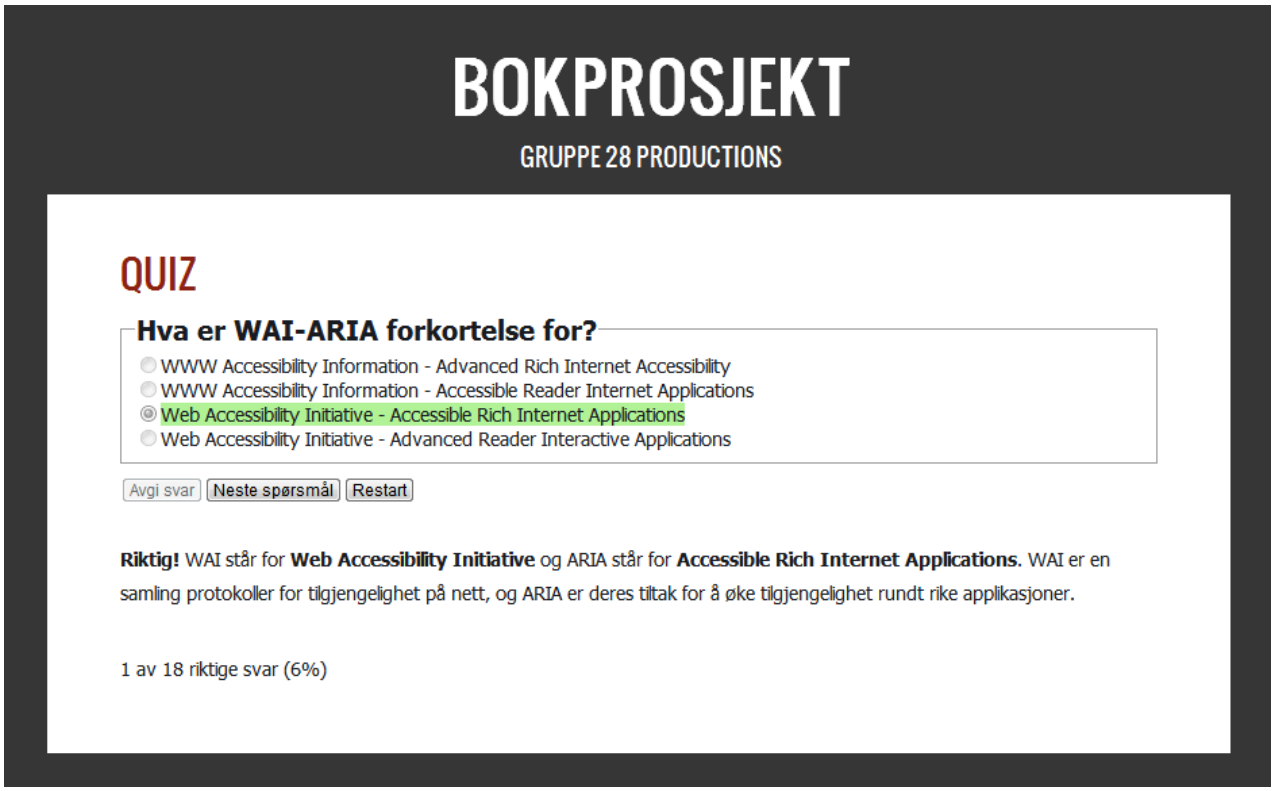
Selv om grafen viser at resultatet med tilbakemelding på slutten av quizen viste seg å være best, vurderte vi det til at det er større sjanse for at tilbakemeldingen faktisk blir lest om den presenteres underveis (vår quiz vil tross alt ikke utføres under strengt kontrollerte forhold).

Får å gå videre til neste spørsmål må brukeren trykke på "Neste spørsmål"-knappen. Dette gjøres helt til quizzen er gjennomgått og man får tilbakemelding om det endelige resultatet.

For å motivere brukerne til å oppnå en god score i quizzen får man tre forskjellige premier basert på den totale poengsummen. Om man ikke får toppscore oppfordres brukeren til å prøve på nytt.

Designet til quizzen stiles gjennom den vanlige CSS-filen som brukes på de øvrige sidene. Forøvrig er det også brukt visuelle effekter fra jQuery biblioteket for å få programmet til å fremstå som mer engasjerende. Effektene som er tatt i bruk er av typen fadeIn() og fadeOut().

Slik ser hele quizapplikasjonen ut i bruk:


BOKPROSJEKT
GRUPPE 28 PRODUCTIONS
QUIZ
Hva er WAI-ARIA forkortelse for?
WWW Accessibility Information - Advanced Rich Internet Accessibility
WWW Accessibility Information - Accessible Reader Internet Applications
Web Accessibility Initiative - Accessible Rich Internet Applications
Web Accessibility Initiative - Advanced Reader Interactive Applications
Avgi svar
Neste spørsmål
Restart
Riktig! WAI står for Web Accessibility Initiative og ARIA står for Accessible Rich Internet Applications. WAI er en samling protokoller for tilgjengelighet på nett, og ARIA er deres tiltak for å øke tilgjengelighet rundt rike applikasjoner.
1 av 18 riktige svar (6%)

5. Avslutning og konklusjon

5.1 Konklusjon

Så godt som alle bedrifter i dag har egne IKT-løsninger, og flertallet av disse har vært i drift lenge før det ble stilt spørsmål om tilgjengelighet. Situasjonen i dag gjør det vanskeligere for assisterende teknologi å holde følge med utviklingen, og det er på sin plass med et nærmere samarbeid mellom produsenter av assisterende teknologi, webutviklere og webdesignere. Dette er noe alle må ta del i fra starten av et utviklingsprosjekt og ikke bare overlates til slutt.

Gruppen sitter nå igjen med et produkt som vil fungere som Norges første guide til WAI-ARIA på norsk og på web. Dette skal være en tilgjengelig nettside formet etter WCAG 2.0 og WAI-ARIA. Dette er en nettside laget for studenter, web-designere eller andre som interesserer seg for universell utforming og ønsker å lære om den nye W3C-standarden, WAI-ARIA som ennå er under utvikling.

Det må bli tatt i betraktning at nettsiden er et læreverk om et snevert og fortsatt ukjent tema, så det forventes i utgangspunktet en generell bakgrunnskompetanse av brukeren, hvertfall innen HTML og helst litt innenfor JavaScript, for å forstå alt av innhold.

Gruppen ønsket å holde seg til prinsipper om hvordan informasjon kan gjøres lettere tilgjengelig. Mange programmeringsguider fremstår som veldig utilgjengelige med overdreven bruk av fagterminologier og enten tungt eller lite pedagogisk innhold. Gruppen har med tanke på dette valgt å benytte et enkelt og lett kommunikasjonsspråk tilpasset de fleste brukergrupper.

Nettsiden har blitt utformet med WCAG og WAI-ARIA først og fremst for å øke tilgjengeligheten men også fordi dette er en nettside som omhandler universell utforming og skal derfor være et eksempel til budskapet.

I forhold til brukertestene gruppen har kjørt på vårt testpanel viser nettsiden seg å være lett navigerbar og innholdet viser seg å være forståelig. Problemene oppsto ved bruk av skjermleser og navigering via tastatur hvor noen av testpersonene hadde lite erfaring med tastaturnavigering og ennå mindre erfaring med bruken av skjermlesere. Kritikk ble opplyst under testingen og spørreskjemaet og vel motatt for mulige endringer.

Gruppen fikk dessverre ikke testet siden med et testpanel som er vant med skjermlesere. Men vår oppdragsgiver Morten Tollefsen som er blind har testet nettsiden med sin JAWS skjermleser. Morten klarte fint å orientere seg uten særlige problemer.

Nettsiden er utformet med maler på alt av nettsidens innhold for å lett kunne produsere å legge inn mer informasjon og demoer. Gruppen håper at nettsiden kan videre utvikles til å bli enda større og mer informasjonsrik.

Gruppen håper at denne nettsiden kan være til hjelp for nyttig læring og forståelse av WAI-ARIA i fremtiden. Med tanke på at dette er et åpent prosjekt føler gruppen at de ikke bare har bidratt med dette for MediaLT sin grunn, men og for å fremme tilgjengligheten av WAI-ARIA og universell utforming for alle som har lyst til å lære.

5.2 Fremtidige muligheter og endringer

Nettsiden er laget slik at den kan utvides med flere demonstrasjoner og undermenyer. Vi har opprettet to maler med php og ferdig css, som man enkelt kan fylle i med mer stoff og dermed beholde det samme utseendet. Man kan lett endre spørsmål og rekkefølge på aria-quizen ved å legge til flere i et array i *quiz.js*. Memoryspillet kan økes med flere lyder i *memory.js* og flere knapper i *memory.html*.

Memoryspillet er i første omgang tilpasset synshemmede og seende som memorerer plassering og lyd. Det er likevel fullt mulig å inkludere hørselshemmede eller brukere uten høyttalere. Måten dette kan løses på er å legge til farger eller bilde under knappene

som må memoreres i tillegg. Grunnen til at vi ikke har prioritert dette er at spillet i utgangspunktet skulle være et lyd-spill, og spillets vanskelighetsgrad vil reduseres i det man får både lyd og farge. En annen forbedring av spillet vil være å inkludere en stoppeklokke slik at man måle sitt resultat. (Dette kolliderer forøvrig med WCAGs retningslinjer om å alltid tilby alternativer til tidsbasert media, men man skal jo ha det gøy også).

Vi har også testet nettsiden uten CSS for å se hvordan den fremstår for en skjermleser og resultatet opplevdes fortsatt som veldig logisk. Det er godt mulig å endre CSS på siden, men dette bør gjøres av noen med erfaring. Ved endring er det viktig å huske på å ikke "låse" designet slik at man fratar brukeren muligheten til å bestemme hvordan siden skal se ut på sin nettleser.

Testdokumentasjon

Innholdsfortegnelse

Innholdsfortegnelse

1. Målet med testingen

Testdokumentasjonen fokuserer på to områder, funksjonell testing og brukertesting. Denne delen av rapporten gir en innføring i testmetodikken og en sammenfatning av testresultatene.

Brukertestingen er ment som et verdifullt hjelpemiddel til å forstå brukernes oppførsel og kunne tilpasse nettsiden i tråd med brukernes forventninger. Dette er den beste måten å optimalisere designet på. Testing gir en muliget for å se produktet sitt med nye øyne og oppdage svakheter som man selv aldri hadde funnet.

Funksjonell testing er ment for å oppdage problemer i kode o.l. og en dokumentering av testresultater vil også være gunstig å ha for dem som skal ta nettsiden i bruk senere.

2. Funksjonell testing

2.1 Quizapplikasjon

2.1.1 Testing av resultatfelt med skjermleser

Formålet med denne testen er å sjekke om det oppstår problemer med fjerning og vising av resultatfeltet når man bruker skjermleser.

Chatapplikasjonen ble åpnet gjennom Google Chrome på en testmaskin mens NVDA kjøres i bakgrunnen. Brukeren velger et svaralternativ og trykker på “Avgi svar”-knappen og observerer hva som skjer når resultatfeltet kommer opp. Eventuelle problemer blir notert.

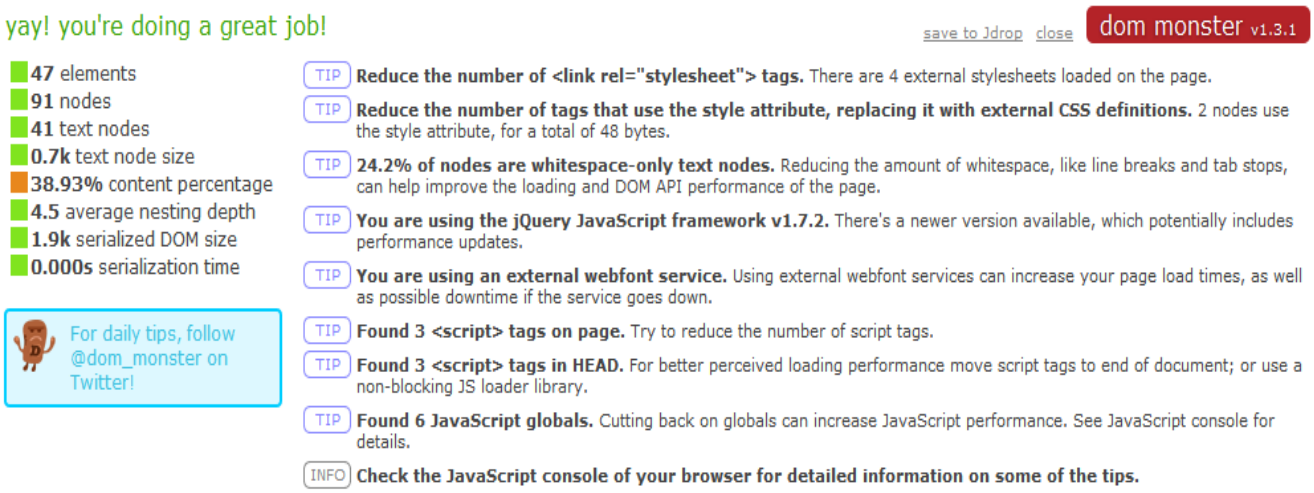
Resultat

Resultatfeltet fungerer tilfredstillende.

2.1.2 Testing av quizens ytelse med DOM Monster

Formålet med denne testen er å se om det kan gjøres forbedringer i forhold til ytelsen til quizen. Testen kjøres ved hjelp av et bokmerke som laster inn et skript. Skriptet vurderer hvor bra ytselsen er på siden, og hvordan man kan gjøre forbedringer i koden.

Resultat



Dette resultatet førte til at jQuery UI ble fjernet som et script fordi det aldri ble brukt (rest etter utvikling). Noen ubrukte skrifttyper som lastes inn fra "Google webfonts" ble også fjernet. Dette gjør i tillegg koden lettere å lese og forstå.

2.2 Chatapplikasjon

2.2.1 Testing av applikasjonens liveregioner med NVDA

Formålet med denne testen var å finne ut om en skjermleseren NVDA ville fange opp og lese oppdateringer (online- og offlinestatus og nye meldinger).

Chatapplikasjonen ble åpnet gjennom Google Chrome på en testmaskin mens NVDA kjøres i bakgrunnen. Radioknappene for å velge liveregionattributt settes til "assertive". Testpersonen noterer ned resultatene etter ett minutt med testing.

Resultat

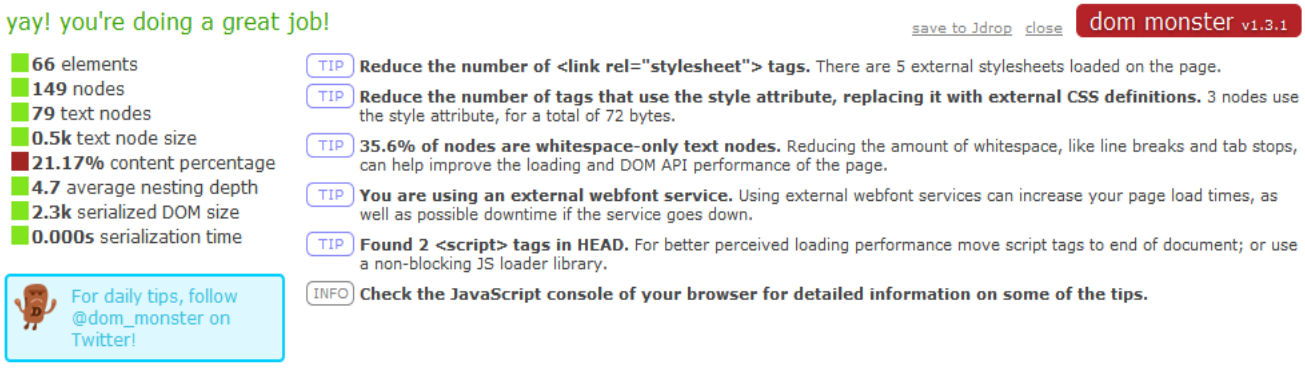
Liveregionene fungerer som de skal, og alle nye meldinger blir lest opp av skjermleseren.

Selv om liveregioner fungerer tilfredsstillende har vi oppdaget at mengden informasjon som genereres er en utfordring å holde oversikt over. Lesehastigheten er også noe som bør tas stilling til. Om lesehastigheten er lavere enn meldingene som kommer opp kan det fort opparbeide seg en kø med tekst som kan gjøre det vanskelig å forstå sammenhengen mellom alle de forskjellige oppdateringene.

2.2.1 Testing av ytelse med DOM Monster

Formålet med denne testen er å se om det kan gjøres forbedringer i forhold til ytelsen til quizen. Testen kjøres ved hjelp av et bokmerke som laster inn et skript. Skriptet vurderer hvor bra ytselsen er på siden, og hvordan man kan gjøre forbedringer i koden.

Resultat



De samme endringene som ble utført i 2.1.2 ble også utført her. I tillegg ble script-elementene flyttet til slutten av dokumentet, rett før body-elementets slutt-tag. Dette gjør

at siden ikke trenger å vente på skriptene før HTML-koden blir tolket av nettleseren. Script-elementene ble også flyttet i quiz-applikasjonen. Denne endringen ga en merkbar effekt i begge applikasjonene.

2.3 Memory med lyd

2.3.1 Testing av memoryspill med NVDA

Memoryspillet ble testet i Google Chrome med NVDA. Skjermleseren leser ut rad og kolonne på knappene og spiller lyd samtidig.

Når skjermleseren skrus på, oppstår det derimot noen underlige effekter, og de er underligere hvis ARIA brukes enn hvis det ikke er i bruk. Vi fant ut at det kunne vært fint å vise fram dette, ved å la halvparten av knappene være ARIA-kodet, den andre halvparten ikke. I de to øverste radene har knappene ikke ARIA-koding, og skjermleseren (NVDA) leser opp posisjon og tilstand på denne formen ved hhv. ikke funnet og funnet knapp:

"Ikke funnet 4 av 4 rad 2 kolonne 4. Ikke funnet knapp."

"Funnet 4 av 4 rad 2 kolonne 4. Funnet knapp"

Ikke like bra som det burde være, men forståelig, i hvert fall etter noe tilvenning.
Med ARIA-koding, i de nederste radene, leses det opp slik:

"Ikke funnet 1 av 4 rad 3 kolonne 1. Ikke funnet trykknapp ikke avkrysset."

"Funnet 1 av 4 rad 3 kolonne 1. Funnet trykknapp trykket ikke avkrysset"

Her blir det en del smør på flesk. Verst er det imidlertid at ARIA-kodingen gir tilsynelatende selvmotsigende opplysninger; *"Funnet trykknapp trykket **ikke avkrysset**".*
Det kan se ut til at skjermleseren pga. ARIA-kodingen bruker verdien til attributtet "aria-checked", som ikke er i bruk i dette eksempelet. Her nevner vi at dette attributtet ifølge ARIA ikke skal brukes på "button"-elementer[1], så dette problemet kan være en feil i NVDA.

[1] http://www.w3.org/TR/wai-aria/roles#button

2.3.1 Resultat

Det kan se ut til at ARIA gir brukeren litt mer informasjon enn nødvendig i noen tilfeller. For mye informasjon kan lett bli forvirrende. Dette gir oss grunn til å konkludere med at ARIA nøye bør vurderes i hvert tilfelle. Applikasjoner som allerede er tilstrekkelig universelt utformet kan gjerne være brukervennlige nok i seg selv.

2.4 Hovedside

2.4.1 Testing av kontrast

Formålet med denne testen er å forsikre oss om at nettsiden har gode nok fargekontraster for å nå suksesskritieriene 1.4.3 (AA, minimum) og 1.4.6 (AAA, enda bedre) i WCAG-retningslinjene.

Vi ønsker å nå suksesskriterie 1.4.6 som krever følgende:

- Text and images of text have a contrast ratio of at least 7:1.
- Large text (over 18 point or 14 point bold) has a contrast ratio of at least 4.5:1

Testen utføres i Mozilla Firefox med tillegget Juicy Studio: Colour Contrast Analyser.

Under Lumosity Contrast Ratio betyr "(pass at level AAA)" at suksesskriteriet 1.4.6 i WCAG 2.0 er nådd.

Resultat

	Failures
Luminosity Contrast Ratio	0
Difference in Brightness	0
Difference in Colour	0

Element	Parent Nodes	Luminosity Contrast Ratio	Difference in Brightness	Difference in Colour
A	• HTML • BODY • DIV#wrapper • DIV#header-wrapper • DIV#header • DIV#logo • H1	15.13:1 (pass at level AAA)	217 (pass)	659 (pass)
P	• HTML • BODY • DIV#wrapper • DIV#header-wrapper • DIV#header • DIV#logo	15.13:1 (pass at level AAA)	217 (pass)	659 (pass)
A	• HTML • BODY • DIV#wrapper • DIV#menu • NAV • UL • LI.currentMenu	15.13:1 (pass at level AAA)	217 (pass)	659 (pass)
A	• HTML • BODY • DIV#wrapper • DIV#menu • NAV • UL • LI	8.93:1 (pass at level AAA)	191 (pass)	573 (pass)
H1	• HTML • BODY • DIV#wrapper • DIV#page • DIV#page-bgtop • DIV#page-bgbtm • DIV#content	8.93:1 (pass at level AAA)	191 (pass)	573 (pass)

	• DIV.post			
P	• HTML • BODY • DIV#wrapper • DIV#page • DIV#page-bgtop • DIV#page-bgbtm • DIV#content • DIV.post	17.26:1 (pass at level AAA)	228 (pass)	682 (pass)
B	• HTML • BODY • DIV#wrapper • DIV#page • DIV#page-bgtop • DIV#page-bgbtm • DIV#content • DIV.post • P	17.26:1 (pass at level AAA)	228 (pass)	682 (pass)
SPAN class: tag	• HTML • BODY • DIV#wrapper • DIV#page • DIV#page-bgtop • DIV#page-bgbtm • DIV#content • DIV.post • PRE.prettyprint.prettyprinted • CODE	17.02:1 (pass at level AAA)	200 (pass)	582 (pass)
SPAN class: pln	• HTML • BODY • DIV#wrapper • DIV#page • DIV#page-bgtop • DIV#page-bgbtm • DIV#content • DIV.post • PRE.prettyprint.prettyprinted • CODE	21:1 (pass at level AAA)	255 (pass)	765 (pass)

SPAN class: atn	• HTML • BODY • DIV#wrapper • DIV#page • DIV#page-bgtop • DIV#page-bgbtm • DIV#content • DIV.post • PRE.prettyprint.prettyprinted • CODE	11.12:1 (pass at level AAA)	193 (pass)	661 (pass)
SPAN class: pun	• HTML • BODY • DIV#wrapper • DIV#page • DIV#page-bgtop • DIV#page-bgbtm • DIV#content • DIV.post • PRE.prettyprint.prettyprinted • CODE	21:1 (pass at level AAA)	255 (pass)	765 (pass)
SPAN class: atv	• HTML • BODY • DIV#wrapper • DIV#page • DIV#page-bgtop • DIV#page-bgbtm • DIV#content • DIV.post • PRE.prettyprint.prettyprinted • CODE	19.56:1 (pass at level AAA)	225 (pass)	510 (pass)
H2	• HTML • BODY • DIV#wrapper • DIV#page • DIV#page-bgtop • DIV#page-bgbtm • DIV#sidebar • UL	8.93:1 (pass at level AAA)	191 (pass)	573 (pass)

	• LI • DIV#boks			
P	• HTML • BODY • DIV#wrapper • DIV#page • DIV#page-bgtop • DIV#page-bgbtm • DIV#sidebar • UL • LI • DIV#boks	17.26:1 (pass at level AAA)	228 (pass)	682 (pass)
A	• HTML • BODY • DIV#wrapper • DIV#page • DIV#page-bgtop • DIV#page-bgbtm • DIV#sidebar • UL • NAV • LI • H2	8.93:1 (pass at level AAA)	191 (pass)	573 (pass)
A	• HTML • BODY • DIV#wrapper • DIV#page • DIV#page-bgtop • DIV#page-bgbtm • DIV#sidebar • UL • NAV • LI • UL • LI	8.93:1 (pass at level AAA)	191 (pass)	573 (pass)
A	• HTML • BODY • DIV#footer	15.13:1 (pass at level AAA)	217 (pass)	659 (pass)

	• P			

Success Criterion	Level	Minimum Ratio	Large Print Minimum Ratio
1.4.3 Contrast (Minimum)	AA	4.5:1	3:1
1.4.6 Contrast (Enhanced)	AAA	7:1	4.5:1

Difference in brightness	Greater than 125
Difference in color	Greater than 500

Med dette testresultatet kan vi med høy sannsynlighet regne med at brukere som er svaksynte eller har problemer med å skille farger kan bruke WAI-ARIA-guiden og alle dens undersider uten problemer (fordi de bruker samme CSS-fil).

2.5 Klikkespill

2.5.1 Testing av spillet med skjermleser

Formålet med testen var å se om spillet fungerte med en skjermleser i forhold til at skjermleser overstyrer tastaturet med funksjoner som og deaktiverer knapper.

Spillet ble testet ved å spille det i Internet Explorer 9, Firefox 12, Chrome 19 og Opera 11.64.
Spillet fungerer ikke i det hele tatt i IE9.
Opera 11.64 har ikke støtte for Live-regioner så der fungerer det som det gjør uten skjermleser.

2.5.2 Resultat

I Chrome og Firefox fungerer spillet ved å bruke tastene F8, 7, 8, 9, 0, +, \ og Z tastene for å få score. Problemet er at alle tastene utenom F8 blir lest opp når de trykkes på, og dette overskriver hva skjermleserens output. Når gratulasjonsmeldingen kommer opp kan man fortsatt være i full fres med trykkingen og den blir ikke lest opp siden man gjør et trykke-input som kansellerer meldingen. Løsningen vil være å sette en tabindex på gratulasjonsmeldingen for å gi den fokus.

2.6 Testing i nett-/skjermlesere

Vi utførte en testing på hver av demonstrasjonene. En på gruppa har Mac og har testet med Safari og VoiceOver. VoiceOver har ikke god støtte for WAI-ARIA, og i såfall kun i sin egen nettleser Safari. Den leser ikke ut ARIA-status. Oppdragsgiver har testet med JAWS og resten er testet med NVDA. Resultater av dette kan leses under de forskjellige demonstrasjonene på vår nettside www.aria.moo.no.

3. Brukertesting

3.1 Formålet med testen / valg av testpersoner

Etter å ha lest om brukertesting på nett ble vi enige om å utføre en grundig test med 5 testpersoner og følge disse nøye mens de benyttet siden. I følge ekspert på brukertesting, Jakob Nielsen[2], er det lite hensiktsmessig å teste på flere enn 5. Hvis flertallet av disse står fast på samme sted kan man med det samme gå ut i fra at man har lav brukervennlighet og at dette bør endres. Slik slipper man å kaste bort tid på å la hundre testpersoner gjennomføre den samme øvelsen.

En god brukertestingsteknikk er også å la noen som ikke kjenner fagområdet få teste nettsiden for å finne ut om språket er tydelig nok. Hver person ble håndplukket fra forskjellige miljø.

3.2 Testverktøy

Vi opprettet et dokument i Google Docs (Docs Form) for brukerundersøkelsen. Slik kunne vi samle inn informasjon om hver av personene og deres tekniske nivå, og se dette i sammenheng med resultatene. Det ferdig skjemaet ble lagret med en egen link man kan sende ut til brukerne. Svarene ble lagret i Docs Spreadsheet hvor man har mulighet til å sammenlikne svar og lage grafer. (Grafer var uansett ikke så aktuelt for oss ettersom vi ble enige om 5 testpersoner).

3.3 Testmetode

Scenarier

[2] http://en.wikipedia.org/wiki/Usability_testing

I den første delen av testen ble testpersonen bedt om å utføre spesifikke oppgaver for å se om det var problemer knyttet til disse.

1. Orientering og navigering

Bruker blir spurt om å navigere seg gjennom spesifikke topmenylinker og sidemenylinker. Forstår bruker hva som er linker?

- Finn frem til informasjon om Aria?
- Finn frem til informasjon om boken?
- Finn frem til Live Regioner?
- Finn frem til LiveRegions Chat?
- Finn frem tilbake til Forsiden?
- Finn frem til tabindex?
- Finn frem til tabindex demo 1?
- Finn frem til Landemerker?
- Finn frem til Landemerker demo 2?

2. Tilegning av informasjon (forståelse av innholdet)

Bruker velger mellom Landemerker, Tabindex eller Live Regioner (kan velge alle) for å fordype seg og forstå hva det er og hvordan det brukes. Var kodeboksene lette å lese? Ville det vært bedre uten farger i kodeboksen?

3. Fri gjennomgang av nettsiden

Bruker navigerer seg fritt gjennom nettsiden i 15 min for å teste ut det teknisk. Bruker blir spurt om å peke ut eventuelle feil, mangler eller ting brukeren ikke likte ved siden.

4. Gjennomgang av quiz

Bruker går i gjennom quiz om ARIA etter å ha lest om WAI-ARIA.

Under hvert av disse punktene noteres testveileders observasjoner og brukerens tilbakemelding ned.

Etter testen er utført får testpersonen utdelt et spørreskjema for å få tak i eventuell tilleggsinformasjon. Her samles også inn kvantitative data om brukeropplevelsen.

Følgende måleverdier blir brukt i spørreskjemaet:

Yrkesstatus
Nettleser: Internet Explorer / Firefox / Chrome / Opera / Safari / Annen
Hvor god er du i HTML?: 1-7
Hvor god er du i Javascript?: 1-7
Hvor vant er du med skjermleser? 1-7
Hva syntes du om designet? 1-7
Hva syntes du om oppbygningen av siden? 1-7
Hvor lett var det å finne fram til forskjellig innhold? 1-7
Vet du hva WAI-ARIA er etter å ha vært inne på nettsiden?: Ja / Nei
Vet du hva Landemerker (i WAI-ARIA) er etter å ha vært inne på nettsiden?: Ja / Nei
Vet du hva Tabindex er etter å ha vært inne på nettsiden?: Ja / Nei
Vet du hva Live-regioner er etter å ha vært inne på nettsiden?: 1-7
Hvordan syntes du quizzen fungerte?: 1-7
Var spørsmålene lette å forstå?: 1-7
Hva syntes du kan gjøre siden bedre?: Brukeren skriver inn tilbakemelding

3.4 Resultater

Testperson 1 - Salgskonsulent / generell interesse for IT
Testperson 2 - Stipendiat ved UiO / veldig lite kunnskap om IT
Testperson 3 - Bachelorstudent ved HiOA / meget IT-kyndig
Testperson 4 - Blind IT-konsulent med skjermleser (oppdragsgiver) / meget IT-kyndig
Testperson 5 - Bachelorstudent UiO / lite kunnskap om nettutvikling

Vi har lagd en liten rapport for hver av testpersonene. Det kunne vært interessant å gruppert disse mot hverandre, men det ville blitt litt forvirrende å lese ettersom man da mister oversikt over den totale læreprosessen hos hver person.

Testperson 1 - Salgskonsulent

Denne personen er interessert i brukervennlig utforming. Dette var den første testen som ble gjennomført, og fungerte som "pilot"test. Testen ble på grunn av testsituasjonen i hovedsak gjennomført med fri navigering, først og fremst med tanke på å finne svakheter ved testopplegget og nettstedet.

1. Orientering og navigering

Finn frem til informasjon om ARIA? Finn frem til informasjon om boken?

Personen fant fram til informasjonen om ARIA uten problemer.

Finn frem til Live-regioner?

Personen var innom siden mest som en tilfeldighet

Finn frem til LiveRegions Chat?

Personen klikket videre til Liveregioner Chat etter å ha lest om live-regioner.

Finn frem tilbake til Forsiden?

Testpersonen fant flere ganger tilbake til forsiden, uten problemer

Finn frem til tabindex?

Testpersonen finner fort fram til tabindex.

Finn frem til tabindex demo 1?

Testpersonen gikk fort videre til tabindex demo 1.

Finn frem til Landemerker?

Personen var innom siden mest som en tilfeldighet

Finn frem til Landemerker demo 2?

Personen var ikke interessert i landemerker-demoene, og gikk tilbake.

2. Tilegning av informasjon (forståelse av innholdet)

Testpersonen leste informasjon på de fleste sidene. Han oppfattet innholdet som interessant og leservennlig, men ønsket seg mer informasjon om de enkelte temaene.

3. Fri gjennomgang av nettsiden

Testpersonen stusset først litt over hvordan nettstedet skulle navigeres. Toppmenyen ble enkelt forstått, men lenkene på siden ble først etter en liten stund oppfattet som lenker, siden utformingen framsto som vanlig tekst. Testpersonen foreslo å forsøke å bytte om topp- og sidemenyene. Han savnet "alt"-tekst på lenkene. Han syntes også det var litt rart at menyene ikke er til stede på testsidene. Testeren brukte Firefox, og irriterte seg over at "page up" og "-down" fungerte ustabilt på sidene. Han forsøkte etterpå med IE, der de tastene fungerte problemfritt. Testpersonen likte sitt svenske navn.

4. Gjennomgang av quiz

Testpersonen gjennomførte ikke quizen.

Testperson 2 - Stipendiat ved UiO

1. Orientering og navigering

Finn frem til informasjon om ARIA? Finn frem til informasjon om boken?

Brukeren ser lenkene i toppmenyen med en gang.

Finn frem til Live Regioner? Finn frem til LiveRegions Chat?

Bruker finner begge i sidemenyen veldig raskt.

Finn frem tilbake til Forsiden?

Forstår ikke at headingen leder tilbake til forsiden, men sier at han vet at det er vanlig. Finner den i hovedmenyen.

Finn frem til tabindex? Finn frem til tabindex demo 1? Finn frem til Landemerker?

Finn frem til Landemerker demo 2?

Ingen problemer. Begynner bli litt kjedelig.

2. Tilegning av informasjon (forståelse av innholdet)

- Brukeren velger tabindex. Har lite bakgrunnskunnskaper om HTML og syns de lange kodeboksene ser skremmende ut. Orker ikke bruke tid på å forstå det noe særlig.

- På tabindex-demo 2 klikker ikke brukeren på knappene fordi han tror det er det samme som tabindex-demo 1. Teksten sier man skal klikke men kan kanskje gjøre dette litt mer tydelig.
- Liker ikke ordet å "tabbe". Antar at det kanskje er IT-språk.
- Skjønner ikke tabellen i siden og hvordan JavaScript får fokus.

3. Fri gjennomgang av nettsiden

- Skjønner ikke at overskriftsnivå 1 i sidemenyen er en lenke, tror det er en overskrift og går istedet rett på den første landemerkedemoen. Denne fremstår derfor underlig. Syns likevel innholdet i demoen er morsomt men forstår ikke at den bør testes med skjermleser. Klikker på lenkene men disse leder ingen vei. Det er tydelig at det bør opplyses på selve siden hvordan denne helst skal brukes: at man skal oppleve hvordan en skjermleser presenterer innholdet med og uten landemerker.
- Nettsiden ble testet med VoiceOver og bruker opplyst om at denne likevel ikke gjør forskjell på disse to demoene. Kan være aktuelt å nevne dette et sted i teksten. Sidemenyen ble gjort tydeligere med { text-decoration:underline; }.
- Klikkespillet fungerer ikke grunnet innstillinger i nettleseren. Skifter fra Safari til Firefox men skjønner likevel ikke at man skal klikke på tabulator først fordi han ikke leste instruksen ordentlig. Tror også at det er mulig å trykke på hvilken som helst knapp. Instruksene i spillet er nå endret til å forklare hvilke knapper som er aktuelle.

4. Gjennomgang av quiz

-Orker ikke å fullføre hele quizen for han mener den er for vanskelig og at det ikke er noe han kommer til å få til. Får derimot riktig på første spørsmål og liker muligheten til å få det riktige svaret servert med en gang.

Testperson 3 - Student ved HiOA

1. Orientering og navigering

Finn frem til informasjon om ARIA?

Testpersonen fant frem til ARIA med en gang.

Finn frem til informasjon om boken?

Testpersonen fant frem til informasjon om boken med en gang.

Finn frem til Live Regioner?

Bruker gikk inn på chat-eksempel i stedet for hovedsiden til liveregioner.

Forvirrende overskrifter.

Finn frem til LiveRegions Chat?

Brukeren har nå forstått strukturen og går inn riktig.

Finn frem tilbake til Forsiden?

Bruker finner frem til forsiden uten å trykke på tilbakeknappen.

Finn frem til tabindex? Finn frem til tabindex demo 1? Finn frem til Landemerker?

Finn frem til Landemerker demo 2?

Ingen problemer nå. Testpersonen finner fort fram til de riktige linkene.

2. Tilegning av informasjon (forståelse av innholdet)

Testpersonen velger å lese om liveregioner.

Skrivefeil under landemerker “in”

Vil heller vite det tekniske med en gang og er ikke så interessert i å lære om den underliggende teorien. Bare kodeeksemplene gir nok utbytte i seg selv for testpersonen.

3. Fri gjennomgang av nettsiden

Klikkespill

Testperson klager på at det kunne blitt løst på en annen måte. Forvirrende med knappene.

Rart med boksen som beskriver menyen (“demoer”)

Forventer at sidemenyen skal endre seg under hvert toppmeny-element

4. Gjennomgang av quiz

Restart-knapp er kanskje litt for lett å trykke på ved et uhell. Flytt helt til høyre.

Forvirrende role vs. attributt-spørsmål er dårlig formulert. Kan ha en attributt som heter attributt.

Gjør om spørsmål om tabindex så det spør om -1 i stedet for "minusverdi".

Manglende svarforklaring på siste spørsmål.

Vil gjerne vite hvilket spørsmål man er på.

Testperson 4 - blind med skjermleser

1. Orientering og navigering

Finn frem til informasjon om ARIA?

Testpersonen bruker søkefunksjon i skjermleseren og finner fram til menyelementet med en gang. Oppfatter at siden inneholder landemerker, men har sin egen strategi for å finne fram.

Finn frem til informasjon om boken?

Testpersonen fant frem til informasjon om like fort som første element.

Finn frem til Live Regioner?

Testpersonen bruker rundt 10 sekunder på å finne frem til denne pga. forvirring rundt menyelementer med relativt like navn.

Finn frem til LiveRegions Chat?

Testpersonen finner fort frem til chatten.

Finn frem tilbake til Forsiden?

Dette tok ganske lang tid med skjermleser, men avhenger av visse forventninger brukeren har til sidens struktur. Hvordan gjøre dette lettere? Testpersonen hadde heller foretrukket å trykke backspace for å komme tilbake.

Finn frem til tabindex?

Finner fram under fem sekunder.

Finn frem til tabindex demo 1?

Litt forvirring gjør at det tar lenger tid.

Finn frem til Landemerker? Finn frem til Landemerker demo 2?

Testpersonen finner fram til de riktige linkene på under fem sekunder (raskt)

2. Tilegning av informasjon (forståelse av innholdet)

Testpersonen velger å lese om liveregioner.

Landemerker brukes til å dele opp siden i logiske seksjoner

De fire første landemerkene er ikke viktigere enn de andre

Dropp "ikke tilrådelig"

Gjør om på "være en logo eller søkeboks", bruk heller search role

Metainformasjon er ikke riktig under complementary

Fjerne "tabbe seg", unøyaktig

Complementary står ikke rundt comment

Informasjon om hvilke nett- og skjermlesere som støtter hva var utdatert og det ble besluttet å fjerne alt om hvilke versjoner som gjelder, ettersom dette er vanskelig å fastslå. Være mer generell.

3. Fri gjennomgang av nettsiden

Siden burde ha title-elementer for hver side. Foreslått struktur: Tabindex demo 1 > Tabindex < Hovedside. Det viktigste først.

Forvirrende forklaring i tabindex-tabell.

4. Gjennomgang av quiz

- Bruker ønsker selv å endre noen av spørsmålene i quizen.
- Quizen fungerer godt med en skjermleser, men det er uklarhet rundt statusen til knappene - om de er "disabled" eller ikke. Avgi svar burde flyttes slik at leseretningen blir: Avgi svar -> Neste spørsmål -> Restart.

Testperson 5 - Bachelorstudent ved UiO

1. Orientering og navigering

Finn frem til informasjon om ARIA? Finn frem til informasjon om boken? Finn frem til Live Regioner? Finn frem til LiveRegions Chat? Finn frem tilbake til Forsiden? Finn frem til tabindex? Finn frem til tabindex demo 1? Finn frem til Landemerker? Finn frem til Landemerker demo 2?

Testpersonen skjønte seg på menyen med én gang. Hadde litt problemer med overskriftene i menyen.

2. Tilegning av informasjon (forståelse av innholdet)

- Brukeren valgte liveregioner. Bør ha en bedre introduksjon (introduksjonen har en del unødvendig informasjon også). Står ikke hvordan det har sammenheng med skjermleser.

3. Fri gjennomgang av nettsiden

- Teksten om fremdriftslinjen var ikke like lettleselig, litt vanskelig å forstå
- Orddelingsfeil under toggleknapp (overskriften på undersiden). Bruk bindestrek: "toggle-knapp"
- Ikke bruke forkortelsen vha (ved hjelp av) under toggle-knapp
- Klikkespillet var litt vanskelig å forstå med en gang

4. Gjennomgang av quiz

- Likte quizzen veldig godt, men ble skuffet over å ikke få sluttresultatet.
- Vil ha opp det riktige svaret selv om man trykker feil (f.eks. rød bakgrunnsfarge på feil svar og grønn på det riktige).

Spørreskjema

Tilbakemelding

T1: Ingenting å tilføye

T2: Ingenting å tilføye

T3: Endre sidemeny for å gjøre det klarere hva som er headings og hva som er linker.

T4: Fylte ikke inn skjema

T5: GI MEG QUIZ-SCORE I MER ENN ET HALVT SEKUND!!!!1!!!1

Se vedlegg for resten av spørreresultatene i vedlegg.

5. Konklusjon

De funksjonelle testene var nødvendig for å få rettet opp feil i demoene og på nettsiden. Dette kan være fint å ha for de som ev. skal drifte siden videre.

Testpersonene ga oss mye nyttig informasjon. Noe fikk vi endret umiddelbart, slik som lenker som ledet feil vei og andre underlige feil som hadde sneket seg inn. Andre problemer ble vi usikre på hvordan vi skulle løse.

Den første testpersonen påpekte at lenkene til de forskjellige temaene var lite tydelige. Etter det forandret vi dem til å være understreket. På tross av det syntes også noen av de senere testpersonene at det ikke var innlysende at temaoverskriftene er lenker.

Ettersom vi brukte testpersoner med forskjellig IT-bakgrunn var det noen som mente ting burde forklares nøyere og enklere. Den av personene som var mest kyndig syntes at det var for mye informasjon og ville gjerne ha mer kodeeksempler tidlig i teksten. Vi vil gjerne at alle skal kunne henge med på innholdet, og antar at de som ikke trenger så mye bakgrunnsinformasjon uansett vil skumme gjennom de det allerede kan.

Samtlige testpersoner så ut til å like de morsomme delene hvor vi hadde vært litt kreative. Det var også lite kritikk av designen så her følte vi at vi hadde oppnådd god leservennlighet. Hadde vi startet testingen litt tidligere kunne vi også formulert noen grundigere spørsmål. Vi fikk likevel et temmelig godt utbytte av bare å observere brukerne og høre på kommentarer underveis.

Brukertestingen har ikke medført vesentlige endringer i forhold til de opprinnelige sidene, så vi kan være ganske fornøyde med hvordan vi utformet dem fra starten. Noen mindre, men ikke uviktige endringer ble gjennomført.

Kildehenvisninger

The jQuery project. jQuery documentation (2012). Hentet fra http://docs.jquery.com/.
Stack Exchange inc. About StackOverflow (2012). Hentet fra http://stackoverflow.com/about.
Jeffrey Way. 30 Days to Learn jQuery (2012). Hentet fra http://tutsplus.com/course/30-days-to-learn-jquery/.

Schwaber K., & Sutherland J. (2008). Scrum. Hentet fra http://www.scrum.org/storage/scrumguides/Scrum%20Guide%20-%20NO.pdf#view=fit

Freedom Scientific, INC, (2012). Welcome to JAWS Headquarters. Hentet fra http://www.freedomscientific.com/jaws-hq.asp

NV Access, (2011). Welcome to the Home of NVDA. Hentet fra http://www.nvda-project.org/

W3Schools, (2012a). HTML5 Introduction. Hentet fra http://www.w3schools.com/html5/html5_intro.asp

W3Schools, (2012b). CSS3 Introduction. Hentet fra http://www.w3schools.com/css3/css3_intro.asp

Web Accessability in Mind (WebAIM) (2008). WebAIM's WCAG 2.0 Checklist for HTML documents. Hentet fra http://webaim.org/standards/wcag/checklist

Web Accessability in Mind (WebAIM). Using NVDA to Evaluate Web Accessibility. Hentet fra http://webaim.org/articles/nvda/

Web Accessability in Mind (WebAIM). Testing with Screen Readers: Questions and Answers. Hentet fra http://webaim.org/articles/screenreader_testing/

W3C (2008). Web Content Accessibility Guidelines (WCAG) 2.0. Hentet fra http://www.w3.org/TR/2008/REC-WCAG20-20081211/

BLD (Barne- likestillings- og inkluderingsdepartementet) (2012). Lovdata § 11. LOV 2008-06-20 nr 42: Lov om forbud mot diskriminering på grunn av nedsatt funksjonsevne (diskriminerings- og tilgjengelighetsloven). Hentet fra http://www.lovdata.no/all/hl-20080620-042.html#11

Sosial og helsedirektoratet/ Deltasenteret, Tilgjengelige nettsteder (brosjyre), 2006, hentet fra:
http://www.bufetat.no/bufdir/deltasenteret/publikasjoner/Tilgjengelige-nettsteder-13---Oversikt-og-innholdsproduksjon/

OpenAjax Alliance, (2012). Accessibility, hentet fra:
http://oaa-accessibility.org/examples/

Donald F. Evans, (2012). ARIA Best Practices, hentet fra:
http://websiteaccessibility.donaldevans.com/#aria

Codecademy, (2012). Learn to Code, hentet fra:
www.Codecademy.com

Vedlegg

Spørreskjema til brukertest

Test av Bokprosjekt

Dette skjemaet er helt anonymt. En sammenfatning av svarene vil bli innført i rapporten "Testdokumentasjon".

* Required

Yrkesstatus * Student, Postmann, Butikkmedarbeider, Arbeidsledig etc.

Hvilken browser brukte du under testen? *

Hvor god er du i HTML? * 1 er kjempedårlig og 10 er kjempegod

1 2 3 4 5 6 7

Hvor god er du i Javascript? * 1 er kjempedårlig og 7 er kjempegod

1 2 3 4 5 6 7

Hvor vant er du med skjermleser? * 1 er svært lite vant og 7 er svært godt vant

1 2 3 4 5 6 7

Hva syntes du om designet? * 1 er grusomt og 7 er fantastisk

1 2 3 4 5 6 7

Hva syntes du om oppbygningen av siden? * 1 er elendig og 7 er kjempegod

1 2 3 4 5 6 7

Hvor lett var det å finne fram til forskjellig innhold? * 1 er nesten umulig og 7 er lekende lett

1 2 3 4 5 6 7

Vet du hva WAI-ARIA er etter å ha vært inne på nettsiden? *

- Ja
- Nei

Vet du hva Landemerker (i WAI-ARIA) er etter å ha vært inne på nettsiden? *

- Ja
- Nei

Vet du hva Tabindex er etter å ha vært inne på nettsiden? *

- Ja
- Nei

Vet du hva Live-regioner er etter å ha vært inne på nettsiden? *

- Ja
- Nei

Hvordan syntes du quizzen fungerte? * 1 er kjempedårlig og 7 er kjempebra

1 2 3 4 5 6 7

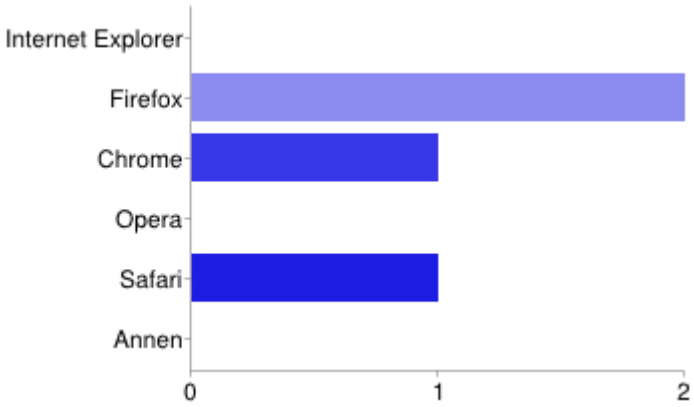
Var spørsmålene lette å forstå? * 1 er uforståelig og 7 er kjempelett

1 2 3 4 5 6 7

Hva syntes du kan gjøre siden bedre? Valgfritt.. Vær ei bekymret, dette er en anonym test: ;-)

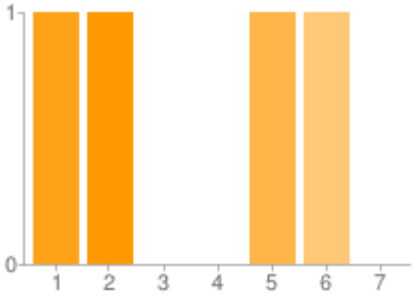
Spørreskjema til brukertest

Hvilken browser brukte du under testen?



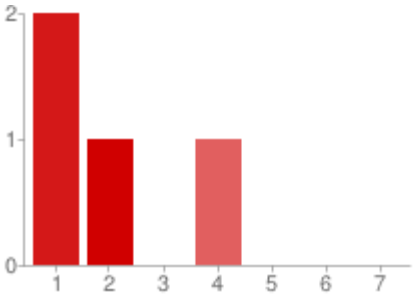
Internet Explorer	0 0%
Firefox	2 50%
Chrome	1 25%
Opera	0 0%
Safari	1 25%
Annen	0 0%

Hvor god er du i HTML?



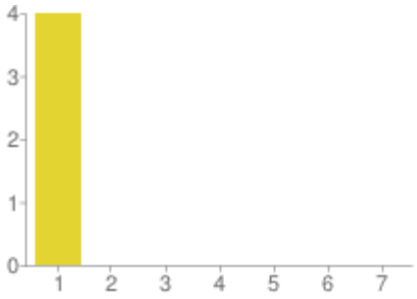
1	1 25%
2	1 25%
3	0 0%
4	0 0%
5	1 25%
6	1 25%
7	0 0%

Hvor god er du i Javascript?



1	2 50%
2	1 25%
3	0 0%
4	1 25%
5	0 0%
6	0 0%
7	0 0%

Hvor vant er du med skjermleser?



1 4 100%

2 0 0%

3 0 0%

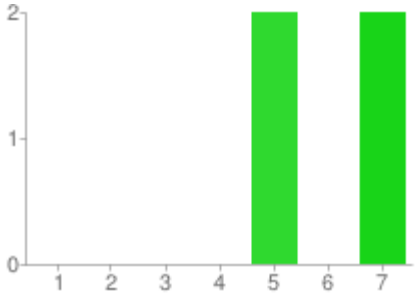
4 0 0%

5 0 0%

6 0 0%

7 0 0%

Hva syntes du om designet?



1 0 0%

2 0 0%

3 0 0%

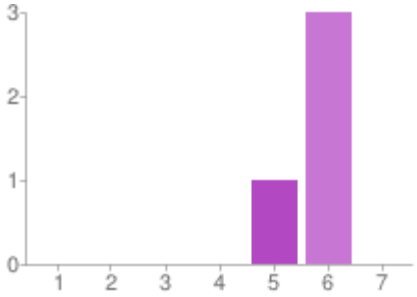
4 0 0%

5 2 50%

6 0 0%

7 2 50%

Hva syntes du om oppbygningen av siden?



1 0 0%

2 0 0%

3 0 0%

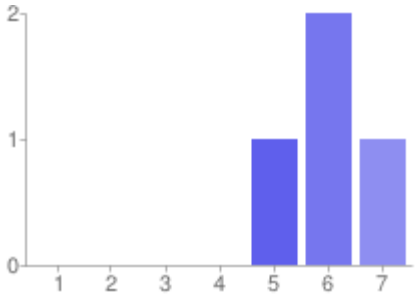
4 0 0%

5 1 25%

6 3 75%

7 0 0%

Hvor lett var det å finne fram til forskjellig innhold?



1 0 0%

2 0 0%

3 0 0%

4 0 0%

5 1 25%

6 2 50%

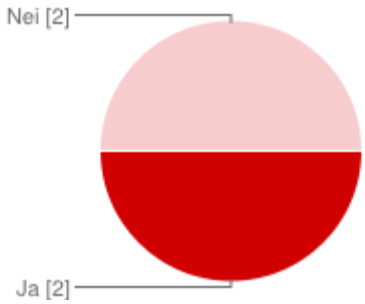
7 1 25%

Vet du hva WAI-ARIA er etter å ha vært inne på nettsiden?

Ja 4 100%

Nei 0 0%

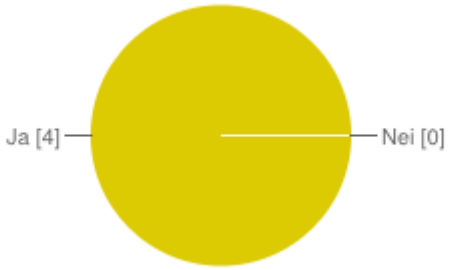
Vet du hva Landemerker (i WAI-ARIA) er etter å ha vært inne på nettsiden?



Ja 2 50%

Nei 2 50%

Vet du hva Tabindex er etter å ha vært inne på nettsiden?



Ja 4 100%

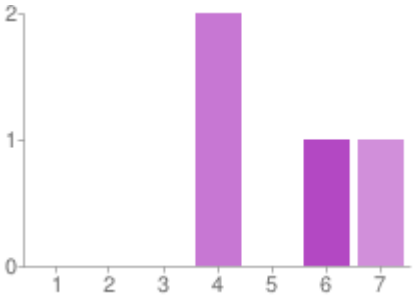
Nei 0 0%

Vet du hva Live-regioner er etter å ha vært inne på nettsiden?

Ja 4 100%

Nei 0 0%

Hvordan syntes du quizzen fungerte?



1 0 0%

2 0 0%

3 0 0%

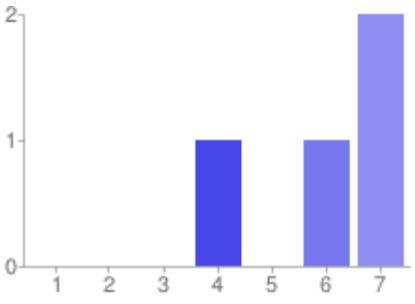
4 2 50%

5 0 0%

6 1 25%

7 1 25%

Var spørsmålene lette å forstå?



1 0 0%

2 0 0%

3 0 0%

4 1 25%

5 0 0%

6 1 25%

7 2 50%

Hva syntes du kan gjøre siden bedre?

Endre sidemeny for å gjøre det klarere hva som er headings og hva som er linker.GI MEG QUIZ-SCORE I MER ENN ET HALVT SEKUND!!!!1!!!1

Brukerveiledning og manual for administratorer

Dette dokumentet redegjør for bruken av vår prosjektnettside.

Brukerveiledning

Nettsidene i dette prosjektet er laget for å vise fram teknikker for å lage tilgjengelige sider ved hjelp av WAI-ARIA og retningslinjene i WCAG, og er rettet mot folk som allerede har en viss kjennskap til produksjon av nettsider. På bakgrunn av dette gis det ikke noen videre forklaring på bruken av tradisjonelle standarder som HTML og CSS, og andre veletablerte metoder som JavaScript og PHP. Unntaket er de tilfellene der WAI-ARIA interagerer direkte med de etablerte språkene og teknikkene. Disse forklaringene er plassert i omtalen av de enkelte eksemplene. Når det gjelder bruken av WAI-ARIA, er den mer detaljert omtalt samme sted, i Produktdokumentasjonen.

WAI-ARIA er et tillegg til HTML-standardene som viser sin anvendelighet best ved bruk av støtteteknologier som skjemlesere, leselister osv., og ved sidenavigasjon med annet enn mus, f.eks. tastatur og talestyring. Ved vanlig visuell lesing og musbruk, vil forskjellen mellom ARIA-kodete sider og sider uten ARIA knapt være merkbar. For å få demonstrert ARIA-bruken, bør nettstedet brukes med en eller annen støtteteknologi.

For den vanlige nettutvikler som skal bruke nettstedet som en innføring i ARIA, og forsøke seg fram med ARIA-koding på sine egne sider, er en skjermleser sammen med tastaturnavigasjon det mest aktuelle alternativet. Her er noen muligheter for den som vil ta i bruk en skjermleser:
Windows har en svært enkel innebygget skjermleser, men den er etter vårt syn ikke god nok til å anbefales. Den mest anerkjente kommersielle skjermleseren er *JAWS* fra Freedom Scientific (http://www.freedomscientific.com/). Den er god, men dessverre svært dyr. Et godt, gratis alternativ er *NVDA* (http://www.nvda-project.org/). Ved bruk av NVDA vil vi anbefale å laste ned og installere Microsoft Speech Platform (MSP) runtime samt en passende tekst til tale-språkpakke (Disse krever Vista eller nyere). Begge deler kan lastes ned fra Microsoft[1]. Da kan man velge MSP i NVDAs innstilling “talesyntese” og en av de

1 http://www.microsoft.com/en-us/download/details.aspx?id=27225 og http://www.microsoft.com/en-us/download/details.aspx?id=27224

installerte språkpakkene i innstillingen "stemmeinnstillinger". MSP har en mye mer naturlig og forståelig stemme enn NVDAs innebygde.
Mac-brukere har det svært enkelt. Skjermleseren *VoiceOver* har vært innebygd i Mac OS siden 2005, og er god nok til de fleste formål.
Det finnes også skjermlesere for **Linux-/*nix**-brukere, men vi har ikke grunnlag for å framheve noen av dem. Vi nevner likevel at den klart mest utbredte, *Orca* (https://live.gnome.org/Orca/), er en del av GNOME-plattformen.
Google Chrome (http://support.google.com/chrome/?hl=no) med skjermleser-tillegget *ChromeVox* utgjør en egen kategori, idet skjermleseren fungerer bare for nettleseren. ChromeVox gir også ekstra navigeringsstøtte i Chrome. Kombinasjonen fungerer ganske godt i både Windows, Mac og Linux.

Fokuset er her på bruken av WAI-ARIA. Derfor er skript, og sidene forøvrig, laget så enkle som mulig uten at funksjonaliteten forsvinner. Skriptene er også tydelig kommentert. Vi anser at de bør være enkelt forståelige for folk med kjennskap til JavaScript og PHP.
Sidene er skrevet i og validert for HTML5, selv om denne standarden ennå ikke er fullstendig fastlagt. De fleste vanlige nettlesere har nå støtte for det vesentligste av denne ikke fastlagte standarden. Samtidig er sidene skrevet med tanke på bakoverkompatibilitet; der det eksisterer (tilnærmet) likeverdige alternativer som er HTML4-kompatible, er de valgt. De fleste sidene er derfor også gyldige HTML4-sider, bortsett fra ARIA-attributtene og, selvfølgelig, at de angir feil DOCTYPE. Siden nettlesere stort sett, i tråd med anbefalingene i HTML(4)[2], ignorerer ukjente attributer, og ukjent/manglende DOCTYPE får de fleste (alle?) nettlesere til å bruke den mest kompatible modusen de kjenner til, vil de fleste sidene vises helt problemfritt også i nettlesere som kun støtter HTML4-standarden. Man kan ikke forvente at disse eldre nettleserne drar nytte av ARIA-attributtene, men skjermlesere som brukes med dem kan kanskje gjøre det.

Selv om alle de mest utbredte nettleserne nå har støtte for WAI-ARIA, er graden og vellykketheten av denne støtten ganske varierende. Utviklingen av prosjektet har vist at Firefox har den generelt klart beste støtten for WAI-ARIA, selv om den ikke er fullkommen. Hvis noen av eksemplene ikke virker som de skal i en annen nettleser,

2 http://www.w3.org/TR/html401/appendix/notes.html#notes-invalid-docs

anbefaler vi å forsøke Firefox. Alle eksemplene her fungerer godt, eller i hvert fall delvis i denne nettleseren (versjon 10.0.x ESR[3]).

Noen av eksemplene har vist seg følsomme for hvilke nettleser-plugins som er installert og i bruk. Hvis et eksempel som kanskje bruker/vil bruke en plugin, feiler, kan det være verdt et forsøk å (de)aktivere, (av)installere, oppdatere eller erstatte plugin-en. Alle sider som innholder lydelementer eller levende bilder (typisk med en "embed"- eller "object"-tag) kan være utsatt for plugin-problemer. Dette gjelder alle nettlesere, og også når problemer som ikke virker direkte tilknyttet multimedia-innholdet inntreffer.

Brukerveiledning for administratorer

Vi forutsetter at nettstedadministratorer har kunnskap om oppsett og drift av www-tjenere, eller vet hvordan nettsider plasseres hos sin tjenerleverandør. Hvis du er usikker på hvordan det gjøres, vil vi be deg studere dokumentasjonen for webtjeneren din, eller ta kontakt med din tjenerleverandør.
Nettstedet leveres som en pakke som kan installeres på en hvilken som helst www-tjener med PHP-tolker. Alle moderne www-tjenere har en PHP-tolker innbygd, eller kan utvides med en. Sjekk at din www-tjener har installert og aktivert PHP-tolkeren. Hvis du bruker en ekstern leverandør, undersøk om den tillater kjøring av PHP-skript, og be den eventuelt om å aktivere kjøring av PHP for dine sider.

Alle interne lenker i prosjektet er relative, slik at pakken kan plasseres i en vilkårlig valgt katalog på www-tjeneren, uten videre tilpasning. Hvis det ønskes lenker fra prosjektet til virksomhetens øvrige sider, må de legges til. Vi forutsetter at nettstedadministratorer har kunnskap om hvordan det gjøres.

3 http://www.mozilla.org/en-US/firefox/organizations/faq/