Att arbeta enligt webbdirektivet -en sammanställning

EU-direktivet som kom under 2016 och som gäller bl. a. offentliga myndigheters webbplatser och mobila applikationer är i stora delar en vidareutveckling av de riktlinjer man inom offentlig sektor arbetat utifrån sedan början av 2000-talet. Genom att implementera detta regelverk (WCAG) bygger man inte bara en applikation som är tillgänglig för alla, utan man värdesäkrar även sin investering genom att man säkerställer bakåt- och framåt kompatibilitet samt att html-koden är felfri.

Att från början arbeta för att följa krav och riktlinjer är enligt mig enklare och billigare än att i efterhand försöka införa ändringar för att uppfylla kraven.

1/ Historik

Under många år utnyttjade man webben för att hämta information, den användes ofta som ett alternativ till trycksaker, något statiskt som man inte interagerade med.

Webbläsare gav ett begränsat stöd för att skapa samband mellan design och kodelement (CSS), man utgick ifrån vad som praktiskt var möjligt utifrån för att skapa ett material som såg hyggligt strukturerat ut. Under lång tid hade presentationen lite eller inget att göra med innehållets semantik och dess möjlighet att spegla innehållet och funktionaliteten man byggde.

Genom att alltfler med tiden kunde skapa innehåll på webben, och då sociala nätverk växte fram, ökade kraven på att fler skulle kunna ta del av tekniken. Behoven av standardisering tillkom för att säkerställa att man utvecklade en gemensam grund för att såväl skapa som tolka material för webben på ett likartat sätt.

För att tillfredsställa detta behövdes att man skapade riktlinjer som möjliggjorde en standardisering och som kunde leda ”leda webben till sin fulla potential”. Detta kunde enbart ske genom att succesivt utveckla protokoll och riktlinjer som på sikt kunde säkerställa webbens långsiktiga tillväxt.

1994 startar ett projekt inom det s.k. World Wide Web Consortium (W3C:s) som är ett internationellt samfund med syftet att utveckla öppna webstandarder där huvudmålet är att göra nyttan med web ”tillgänglig för alla människor, oavsett hårdvara, mjukvara, nätverksinfrastruktur, mänskligt språk, kultur, geografisk plats eller fysisk eller mental förmåga”.

Man söker säkerställa tillgänglighet genom välkonstruerade design principer.

Informationssamhälle öppet för alla

“Sverige skulle vara det första landet med ett informationssamhälle öppet för alla.”

Statskontoret fick 2002 i uppdrag att främja utvecklingen av den s.k. 24-timmarsmyndigheten och skapade vägledningen 24-timmarswebben (Statskontoret 2002:13). Under hösten 2003 initierade Statskontoret tillsammans med svenska W3C-kontoret och Handikappombudsmannen (HO) ett samarbete för att vidareutveckla vägledningen.

Vägledningen ”24-timmarswebben” var en svensk publikation som innehöll riktlinjer för utveckling av webb och e-tjänster i offentlig sektor. Vägledningen förvaltades och utvecklades av VERVA som lades ned 2008.

2/ Att tolka och arbeta enligt webbdirektivet

Jag har skrivit följande text som genom konkreta exempel ska kunna förenkla det löpande arbetet vid uppbyggnaden av kod som direkt kan skapas enligt direktiven.

I materialet utgår jag ifrån texter och information från lagförslag “Genomförande av webbtillgänglighetsdirektivet (Ds 2017:60) “, samt från information på webbplatserna: PTS, post och telegrafstyrelsen (se https://webbriktlinjer.se/) samt Myndigheten för digital förvaltning (https://www.digg.se/).

Juridiska krav på tillgänglighet

“Offentliga sektorn har ett särskilt ansvar för tillgänglighet eftersom det är en del av arbetet med att säkerställa mänskliga rättigheter. Alla statliga myndigheter ska enligt förordning (2001:526) om de statliga myndigheternas ansvar för genomförandet av funktionshinderspolitiken verka för tillgänglighet, göra inventeringar och ta fram handlingsplaner. Även andra organisationer bör koppla sitt tillgänglighetsarbete till socialt ansvarstagande, dels för att förebygga diskriminering och att nå alla sina intressenter, dels för att säkerställa en socialt hållbar utveckling.”

WCAG innehåller fyra principer för tillgänglighet:

1. Ska vara möjlig att uppfatta, vilket innebär att information och komponenter i ett användargränssnitt ska presenteras för användare på ett sätt som de kan uppfatta; en text ska exempelvis kunna konverteras till stor stil, punktskrift eller tal.
2. Komponenterna ska vara hanterbara. Detta innebär bl.a. att det ska gå att navigera på en webbplats enbart med hjälp av ett tangentbord.
3. Ska vara begriplig. Detta uppnås t.ex. genom att sidans språk anges i HTML-koden, så att skärmläsare, som används av personer med synnedsättning, använder ett korrekt uttal. Genom att använda semantiskt korrekt kod kan denna tolkas av skärmläsare, som används av personer med synnedsättning. Fundamentalt är då att sidans språk anges i HTML-koden.
4. En webbplats innehåll ska vara robust nog för att kunna tolkas på ett pålitligt sätt av ett brett spektrum av olika användarprogram, inklusive hjälpmedel.

För att uppnå dessa mål finns tolv riktlinjer som representerar grundläggande mål för att göra innehåll mer tillgängligt.

Riktlinjerna finns angivna i tillgänglighetskraven "WCAG 2.1” Det finns tre nivåer för hur väl en webbplats tillgodoser tillgänglighetskraven: A, AA och AAA.

Nedan följer en lista av tillämpningar för att uppnå A och AA standard vid utveckling av applikationer:

(Baseras på https://webbriktlinjer.se/wcag/?checklista). Varje regel har förklaringar via länkar till https://webbriktlinjer.se och/eller https://www.wcag.org/.

I listan finns enbart de kraven som är märkta A och AA, dvs. de kraven som ingår i webbdirektivet.

Några exempel på kritiska detaljer:

Validera koden löpande genom projektet: 4.1.1 (A) Ha som rutin att löpande testa koden enligt https://validator.w3.org/. Därmed uppnås och bibehålls en god kodsemantik.
Genom att löpande testa att möjligheten att navigera med tangentbordet försäkrar man sig om att den viktiga regeln 2.4.3 (A) följs.
Många applikationer interagerar med sina användare via formulär. Se till att alla formulär i en applikation fungerar på lika sätt, följ regel: 1.3.1 (A)

1.1.1 (A)	R115	Beskriv med text allt innehåll som inte är text • Välj detaljnivå efter användarens behov • Undvik upprepningar genom att provlyssna **Utdrag från WCAG-standarden** Riktlinje 1.1 Textalternativ: Tillhandahåll alternativ i form av text till allt icke-textbaserat innehåll så att det kan konverteras till format som användarna behöver, till exempel stor stil, punktskrift, tal, symboler eller enklare språk. Använd alt-attribut för att ge en icke visuell beskrivning av bilder på webben. Vid längre beskrivning används aria-describedby. Exempel: <img src="ladymacbeth.jpg" alt="Lady MacBeth" aria-describedby="p1"> This painting dates back to 1730 and is oil on canvas. It was created by Jean-Guy Millome, and represents ... https://webbriktlinjer.se/riktlinjer/115-textalternativ/#utdrag-fran-wcag-standarden	PRIO 1
1.3.1 (A)	R121	Ange i kod vad sidans olika delar har för roll • R105. Skapa rubriker med h-element • R104. Använd rätt html-element när ni gör listor	PRIO 1

		• Namnge formulärfält med kopplade label-element. Se R55. Skapa tydliga och klickbara fältetiketter. • R98. Skriv rubriker till tabeller • R101. Markera obligatoriska fält i formulär • Betona innehåll med elementet em och inte bara kursivering, eftersom det inte går att kursivera skärmläsarens tal. • Använd WAI-ARIA för sådant som inte går att uttrycka med vanlig html. **R104. Använd rätt html-element när ni gör listor. UL, OL, DL** *En ordnad lista används när punkterna behöver komma i rätt ordning. Det är inte viktigt i en oordnad lista.* **UL:** oordnade listor – punktlistor **OL:** ordnade, numrerade, listor **DL**: definitionslistor. **<P>** är stycke. Och ** ** används inte för att skapa extra mellanrum mellan rader. **Tabell** används enbart som definition tabell. Aldrig för att låsa kodelement i position. **Formulär** Komplexa formulär delas upp i sektioner (fieldset) där varje sektion ges en etikett (legend) som beskrivning för den grupp av inmatningsfält som innefattas. Legend kan formuleras på ett sätt som stödjer interaktion som utgår från en skärmläsare, döljas visuellt från skärmvyn. Genom att i kod (med hjälp av for-attributet på label-elementet) koppla etiketten till fältet kan användare markera fältet även genom att klicka på etiketten, vilket ökar den klickbara ytan. Genom kopplingen blir det även möjligt för en person som saknar en visuell presentation att veta vilken etikett som hör till vilket fält, eftersom skärmläsare läser upp etiketten när fältet får fokus. Som stöd kan även attributen “aria-labelledby” eller “title” användas. <fieldset> <legend>Output format</legend> <input type="radio" name="format" id="txt" value="txt" checked> <label for="txt">Text file</label> <input type="radio" name="format" id="csv" value="csv"> <label for="csv">CSV file</label> […] </fieldset> Design	

		Textfält och rullgardinsmenyer: sätt etiketten ovanför eller till vänster om fältet. • Radioknappar och kryssrutor: sätt etiketten till höger om knappen eller rutan. • Vänsterjustera fältetiketterna. Använd WAI-ARIA för sådant som inte går att uttrycka med vanlig html Exemplet gäller en "non-native" checkbox. <li tabindex="0" class="checkbox" checked> Receive promotional offers “While this works fine for sighted users, a screen reader will give no indication that the element is meant to be a checkbox, so low-vision users may miss the element entirely. Using ARIA attributes, however, we can give the element the missing information so the screen reader can properly interpret it. Here, we've added the role and aria-checked attributes to explicitly identify the element as a checkbox and to specify that it is checked by default. The list item will now be added to the accessibility tree and a screen reader will correctly report it as a checkbox.” <li tabindex="0" class="checkbox" role="checkbox" checked aria-checked="true"> Receive promotional offers 	
1.3.2 (A)	R122	Presentera innehållet i en meningsfull ordning för alla • Testa läsordningen genom att granska en sida av varje sidtyp med några skärmar i olika storlek och genom att lyssna igenom innehållet med en skärmläsare. • Se till och testa även att läsordningen är logisk i dokument som inte är html (pdf, word med mera). *Anpassningsbart: Skapa innehåll som kan presenteras på olika sätt (exempelvis med enklare layout) utan att information eller struktur går förlorad. Innehållet ska struktureras på ett sådant sätt att det är begripligt även med css bortkopplat.*	PRIO 1
1.3.3 (A)	R123	Gör inte instruktioner beroende av sensoriska kännetecken • Använd gärna sensoriska kännetecken (färg, form med mera) för instruktioner eftersom det kan vara en effektiv metod för att underlätta för användarna,	

		inklusive personer med kognitiva begränsningar men kom ihåg att komplettera informationen så att alla kan förstå den.	
1.3.4 (AA)	R153	Se till att allt innehåll presenteras rätt oavsett skärmens riktning • Använd bara i undantagsfall de tekniker som finns för att låsa skärmens riktning (exempelvis Screen Orientation API). • Se till att allt innehåll presenteras oavsett skärmens riktning.	
1.3.5 (AA)	R154	Märk upp vanliga formulärfält i koden • Använd attributet autocomplete på inmatningsfält. • Beskriv förväntat innehåll med attributet autocomplete, om det finns en standardiserad benämning. • Ange autocomplete=”off” om det gäller känslig information eller om servern erbjuder ordförslag.	
1.4.1 (A)	R124	Använd inte enbart färg för att förmedla information Komplettera färgskillnader: • i text med understrykning, ram, fetstil, kursivering eller annat teckensnitt. • med ikoner. • med mönster i diagram och kartor för att särskilja ytmarkeringar. • med beskrivning i text. • med semantisk kodning. Var särskilt försiktig med färgerna grön, röd och brun. Många personer med färgblindhet har svårt att särskilja dessa.	
1.4.10 (AA)	R91	Skapa en flexibel layout som fungerar vid förstoring eller liten skärm • Undvik horisontell scrollning ner till (320 pixlars bredd). • Använd i första hand responsiv design. • Gör en anpassad mobilversion om responsiv design är inte är möjligt. • Även dokument som inte är webbsidor bör kunna presenteras i begränsad bredd. *Exempel:* **Flytande layout ger flexibel spaltbredd** En flytande layout använder spaltbredder i procent för att anpassa sig efter webbläsarfönstrets bredd. Fördelen är att användaren slipper en horisontell rullningslist (scroll bar). Nackdelarna är att extremt smala fönster gör att spalter överlappar varandra, samt att breda fönster ger för långa textrader som är svåra att läsa. **Elastisk layout kontrollerar radlängden** En elastisk layout använder enheten rem för spaltbredder, vilket gör att layouten anpassar sig efter webbläsarens textstorlek. Om användaren ändrar textstorleken ändras också den maximala radlängden, vilket gör att textrader består av samma antal tecken oavsett textstorlek. Fördelen är kontroll över radlängderna. Nackdelen är att ett smalt fönster eller stor text ger en horisontell rullningslist.	PRIO 1

		Hybridlayout blandar flytande och elastisk layout Det går att kombinera flytande och elastisk layout. Då kan man till exempel göra vissa spalter elastiska (bredd i rem) och vissa flytande (bredd i procent). Med hjälp av CSS-egenskaperna för maximal och minimal bredd kan man undvika såväl överlapp som för långa rader."	
1.4.11 (AA)	R156	Använd tillräckliga kontraster i komponenter och grafik • Ge gränssnittskomponenter tydliga visuella gränser. • Gör helst även inaktiva element urskiljbara för alla. • Använd god kontrast för informationsbärande delar av illustrationer och annat grafiskt innehåll, så långt det är rimligt.	
1.4.12 (AA)	R157	Se till att det går att öka avstånd mellan tecken, rader, stycken och ord • Placera text i utrymmen som är tillräckligt stora eller som anpassar sig efter innehållet. • Ta hänsyn till att texter kan ändra storlek om du använder absolut positionering av element som riskerar att krocka med texten.	PRIO 1
1.4.13 (AA)	R158	Popup-funktioner ska kunna hanteras och stängas av alla • Överväg att presentera innehållet på något annat sätt. • Gör det enkelt att ta bort innehållet. • Gör det möjligt att hantera innehållet för alla.	
1.4.3 (AA)	R126	Använd tillräcklig kontrast mellan text och bakgrund • Lita inte på automatisk granskning av kontraster • Överträffa gärna gränsvärdena för kontrast • Överväg att låta användaren välja kontraster	
1.4.4 (AA)	R127	Se till att text går att förstora utan problem • Kontrollera förstoring vid utveckling av stilmallar och sidmallar • Använd relativa mått	
1.4.5 (AA)	R128	Använd text, inte bilder, för att visa text När bilder trots allt behöver innehålla text kan denna ofta göras tillgänglig för alla genom att: • Använda CSS för att placera text i bilden, istället för att ha texten avbildad. • Generera bilden dynamiskt med hänsyn till användarens preferenser för teckensnitt, storlek och förgrund- eller bakgrund. Observera att detta kan kräva en del programmering och att sidan kan behöva innehålla kontroller för att ställa in sådana preferenser. Komplettera med en alt-text. • Skriv alt-text till knappar, logotyper, skärmdumpar och diagram som förmedlar samma information som bilden. • En bild av ett handskrivet brev kan ha antingen en alt-text som återger innehållet eller en kompletterande text med samma funktion.	Design
2.1.1 (A)	R129	Utveckla systemet så att det går att hantera med enbart tangentbordet Testa alla webbplatser och applikationer utan mus och utan pekskärm.	PRIO 1
2.1.2 (A)	R130	Se till att markören inte fastnar vid tangentbordsnavigation	

<table>
<tr><td></td><td></td><td>Testa alla webbplatser och applikationer utan mus och utan pekskärm, och se till att det går att använda alla funktioner som behövs.</td><td></td></tr>
<tr><td>2.1.4 (A)</td><td>R68</td><td>Skapa kortkommandon med varsamhet
Använd kortkommandon sparsamt.
Använd vedertagna tangentkombinationer om sådana finns.
Informera om vilka kortkommandon ni erbjuder.
Gör det möjligt att stänga av eller byta ut kortkommandon som bara består av ett tecken.</td><td></td></tr>
<tr><td>2.2.1 (A)</td><td>R131</td><td>Ge användarna möjlighet att justera tidsbegränsningar
För vissa användare och i vissa sammanhang behövs gott om tid för att använda digitala tjänster. Ge därför användare möjlighet att stänga av, anpassa eller utöka eventuella tidsbegränsningar, om det inte är orimligt för att uppnå sajtens syfte.</td><td></td></tr>
<tr><td>2.2.2 (A)</td><td>R132</td><td>Ge användarna möjlighet att pausa eller stänga av rörelser
Om en animation eller dynamisk uppdatering startar automatiskt och pågår i mer än 5 sekunder ska användaren enkelt kunna undvika den.</td><td></td></tr>
<tr><td>2.3.1 (A)</td><td>R133</td><td>Orsaka inte epileptiska anfall genom blinkande
Maximalt tre växlingar från ljus till mörk eller tvärtom inom en sekund är acceptabelt.</td><td></td></tr>
<tr><td>2.4.1 (A)</td><td>R75</td><td>Erbjud möjlighet att hoppa förbi återkommande innehåll
Skapa genvägar för att hoppa över delar i strukturen till exempel menyn, för att komma direkt till sidans innehåll.
Skapa rubriker med h-element, eftersom skärmläsare låter användarna snabbnavigera med hjälp av sidans rubriker.
Använd WAI-ARIA landmark roles, till exempel main, search, navigation, banner, contentinfo och så vidare. Det gör att användare med exempelvis skärmläsare kan navigera mellan sidans olika delar på ett standardiserat sätt.
Om du använder HTML5, använd strukturelement som main, aside, header, footer och nav för att definiera vilken roll varje del av sidan har.
R68. Skapa snabbkommandon vid behov.
<table>
<tr><th>Funktion/del av sida</th><th>HTML5 strukturelement
(med exempel på aria-label)</th></tr>
<tr><td>Sidhuvud/beskrivning av webbplatsen</td><td><header></td></tr>
<tr><td>Sökfunktion</td><td>Saknas, använd ARIA!</td></tr>
<tr><td>Navigation/meny (kan finnas flera)</td><td><nav aria-label=”huvudmeny”></td></tr>
<tr><td>Huvudinnehåll</td><td><main></td></tr>
<tr><td>Andra delar som står för sig själv. Exempelvis en puff som informerar om annan sida, eller en ruta med fristående information. Kan finnas flera.</td><td><aside></td></tr>
<tr><td>Formulär (dock ej webbplatsens sökfunktion, se ovan).</td><td><form aria-label=”anmälningsformulär”></td></tr>
<tr><td>Sidfot/beskrivning av sidans huvudinnehåll, exempelvis upphovsrätt, användarvillkor</td><td><footer></td></tr>
</table></td><td></td></tr>
</table>

		<table><tr><td>Landmärke som används för att identifiera regioner på en sida. (kan finnas flera, men använd bara för delar som är så viktiga att de behöver finnas i sidnavigation)</td><td><section aria-labelledby=”region1”> <h2 id=”region1”>rubrik för region 1</h2> … Innehåll för region 1… </section></td></tr></table>	
2.4.2 (A)	R135	Skriv beskrivande sidtitlar • Beskriv sidans ämne eller innehåll. • Formulera en titel som går att förstå på egen hand. Det kan till exempel innebära att avsändaren eller webbplatsens namn anges i slutet av sidtiteln. • Titeln bör vara så tydlig och så unik som möjligt utan att bli för lång.	
2.4.3 (A)	R136	Gör en logisk tab-ordning • Kontrollera ordningen utan pekdon. • Säkerställ en logisk ordning med tabindex. 1. Justera ordningen med hjälp av html-attributet tabindex. Det kan anges i html-kod eller ändras dynamiskt med javascript. De presenteras i följande ordning för användaren: Element med tabindex 1, 2, 3 och så vidare i nummerordning. (Om två element har samma tabindex presenteras de i den ordning de förekommer i koden.) 2. Element som saknar tabindex eller har tabindex 0. Dessa presenteras i den ordning de förekommer i koden. Om användaren därefter fortsätter hamnar fokus återigen på det element som ligger allra först i fokusordningen. 3. Element med negativt värde på tabindex plockas bort från fokusordningen. Dit kan användaren alltså inte navigera.	PRIO 1
2.4.4 (A)	R5	Skriv tydliga länkar • Formulera länktext med omsorg. Användaren måste kunna förutse vad som händer vid klick på länken. • Låt sammanhanget och syftet med länken avgöra om den exempelvis ska placeras i brödtexten eller utanför. • Utforma länkar till dokument och länkar till e-post så att användaren får rätt förväntningar. • Länktexten ska vara självförklarande eller förtydligas genom title-text. • Ange i länken om denna öppnas i nytt fönster	
2.4.5 (AA)	R32	Erbjud användarna flera olika sätt att navigera • Erbjud flera olika navigeringsstöd på webbplatsen. • Utgå från användarnas behov och webbplatsens komplexitet när ni väljer navigeringsstöd. • Erbjud en sökfunktion.	
2.4.6 (AA)	R61	Skriv beskrivande rubriker och etiketter • Använd nyckelord ur texten.	

		• Skriv de viktigaste orden först. • Använd aktivt språk, gärna verb. • Gör inte rubrikerna längre än 5-10 ord.	
2.4.7 (AA)	R140	Markera tydligt vilket fält eller element som är i fokus Använd CSS för att tydligt visa vilket element som är i fokus. Exempellösning: #mainnav a:hover, #mainnav a:active, #mainnav a:focus { background-color: #DCFFFF; color:#000066;}	
2.5.1 (A)	R160	Erbjud alternativ till komplexa fingerrörelser • Se till att all funktionalitet går att utföra med flera fingrar även går att utföra med bara ett finger	
2.5.2 (A)	R161	Gör det möjligt att ångra klick • Ge i första hand användaren möjlighet att ångra åtgärden innan nedtryckningen upphör (up-eventet). • Ge i andra hand användaren möjlighet att ångra sig efter up-eventet. • Koppla bara i undantagsfall aktivering till nedtryckning av knappen eller skärmen (down-eventet).	
2.5.3 (A)	R162	Se till att text på knappar och kontroller överensstämmer med maskinläsbara etiketter • Ta reda på vilken maskinläsbar etikett som används för kontrollen. • Se till att den maskinläsbara etiketten matchar den synliga. • Använd bara samma maskinläsbara etikett för flera kontroller om de gör exakt samma sak.	
2.5.4 (A)	R163	Erbjud alternativ till rörelsestyrning • Se till att funktioner som kan hanteras med rörelsestyrning även kan hanteras på något annat sätt. • Gör det möjligt att stänga av rörelsestyrningen.	
3.1.1 (A)	R141	Ange sidans språk i koden • Ange huvudspråk med hjälp av lang-attribut på sidans rot-element.	
3.1.2 (AA)	R142	Ange språkförändringar i koden Ange aktuellt språk med lang-attribut på omslutande element när är språket i ett element är ett annat än sidans huvudspråk.	
3.2.1 (A)	R143	Utför inga oväntade förändringar vid fokusering • Utför bara förändringar (till exempel öppning av fönster eller förändring av värde) när användaren förväntar sig dem.	
3.2.2 (A)	R144	Utför inga oväntade förändringar vid inmatning • Utför bara förändringar (till exempel öppning av fönster eller förändring av värde) när användaren har anledning att förvänta sig dem.	
3.2.3 (AA)	R29	Var konsekvent i navigation, struktur och utformning • Låt gränssnittselement ha samma utseende, funktionalitet och placering på hela webbplatsen.	
3.2.4 (AA)	R146	Benämn funktioner konsekvent • Använd samma termer för återkommande funktioner såsom knappar och ikoner, eftersom vissa användare saknar till exempel layout och formgivning som annars kan användas för orientering.	

3.3.1 (A)	R2	Visa var ett fel uppstått och beskriv det tydligt • Sammanfatta felen och använd en layout som tydligt separerar felmeddelanden från resten av webbplatsens design. • Skriv välformulerade felmeddelanden så ökar chansen att användarna gör rätt från början. • Markera fel och felmeddelanden med WAI-ARIA så att de uppfattas tydligt av användare med hjälpmedel. • Spara det som inte är fel. **Exempel:** Markera fel och felmeddelanden med WAI-ARIA så att de uppfattas tydligt av användare med hjälpmedel. Rekommendationer för tydliga felmeddelanden Sammanfatta felen och använd en layout som tydligt separerar felmeddelanden från resten av webbplatsens design. Skriv välformulerade felmeddelanden så ökar chansen att användarna gör rätt från början. Markera fel och felmeddelanden med WAI-ARIA så att de uppfattas tydligt av användare med hjälpmedel.* Spara det som inte är fel. Sammanfatta felen och använd en layout som tydligt separerar felmeddelanden från resten av webbplatsens design Markera fel och felmeddelanden med ARIA så att de uppfattas tydligt av användare med hjälpmedel För användare som till exempel lyssnar igenom sidan kan det vara mycket svårt att hitta ett fel. Ett sätt att säkerställa att användare med skärmläsare blir informerade om identifierade fel är att markera både fel och felmeddelande i koden, med hjälp av ARIA: Attributet aria-invalid kan sättas dynamiskt på ett formulärelement för att indikera att det innehåller ett fel (till exempel att värde saknas trots att fältet är obligatoriskt, eller att formatet på angiven data är fel). Aria-attributet role med värdet alertdialog kan användas för att indikera med kod att ett felmeddelande presenteras. Då skapas en notifiering som gör att användaren inte missar meddelandet.	
3.3.2 (A)	R55	Skapa tydliga och klickbara fältetiketter • Skriv tydliga och informativa fältetiketter • Koppla ihop fältetikett och inmatningsfält så att även etiketten blir klickbar.	

		• Placera fältetiketterna där användarna lätt ser dem. • Skriv utförliga instruktioner före formuläret, när sådana behövs. • Undvik att göra lösningen beroende av title-attribut och placeholder-texter.	
3.3.3 (AA)	R149	Ge förslag på hur fel kan rättas till • R52. Fyll formulär med kända uppgifter • R57. Låt användarna fylla i information i valfritt format • R101. Markera obligatoriska fält i formulär • R112. Ge ordförslag i sökning och inmatningsfält	
3.3.4 (AA)	R150	Ge möjlighet att ångra, korrigera eller bekräfta vid viktiga transaktioner Erbjud användarna minst en, men gärna fler, av följande skyddsåtgärder: • Möjlighet att ångra sin åtgärd. • Möjlighet att rätta till möjliga fel som systemet identifierat. • Möjlighet att förhandsgranska sina uppgifter och rätta eventuella fel innan åtgärden slutligen bekräftas.	
4.1.1 (A)	R84	Se till att koden validerar • Kontrollera att era mallar för funktioner, tjänster och stilmallar validerar i enlighet med er valda standard. • Kräv att leverantören vid leverans bifogar valideringsprotokoll för samtliga mallar. Mallar som inte validerar bör inte godkännas för leverans, om inte leverantören har acceptabla argument för alla valideringsfel. • Försök att automatisera en regelbunden kodvalidering, eller gör validering till en rutinåtgärd vid all förändring av webbplatsens kod. Det är lätt hänt att tidigare korrekt kod går sönder. Det kan till exempel hända när ni uppdaterar ett tilläggsprogram, när ni infogar en videospelare i ett blogginlägg eller när någon gör ett inlägg i ett kommentarssystem. https://validator.w3.org/ https://jigsaw.w3.org/css-validator/	PRIO 1
4.1.2 (A)	R152	Se till att skräddarsydda komponenter fungerar i hjälpmedel • Använd i första hand standardkomponenter som finns i html. Då uppfyller du automatiskt detta krav. Bara när det finns starka skäl och tillräckliga resurser för test och utveckling bör skräddarsydda komponenter utvecklas. • Vid val av tilläggsprogram eller kodplattformar (till exempel olika javascript-bibliotek) behöver ni undersöka om eventuella komponenter som bygger på dessa plattformar har ett bra stöd för tillgänglighet.	
4.1.3 (AA)	R164	Se till att hjälpmedel kan presentera meddelanden som inte är i fokus • Markera med aria-kod de områden där	