Hvordan kan sterkt synshemmede barn og unge komme i gang med koding?

|  |  |
| --- | --- |
| Skrevet av: | Morten Tollefsen |
| Sist oppdatert: | 24.8.21 |

MediaLT

Jerikoveien 22

1067 Oslo

Tlf: 21538010

E-post: [info@medialt.no](mailto:info@medialt.no)

<www.medialt.no>



Innhold

[1 Forord 4](#_Toc80699002)

[2 Innledning 5](#_Toc80699003)

[2.1 Prosjektmål og milepæler 6](#_Toc80699004)

[2.2 To hensyn å ta 6](#_Toc80699005)

[2.3 Koding eller programmering? 6](#_Toc80699006)

[3 Hvorfor skal synshemmede lære koding? 7](#_Toc80699007)

[3.1 Kreativitet og løsningsfokus 7](#_Toc80699008)

[3.2 Begynn tidlig! 8](#_Toc80699009)

[3.3 Start løpet mot å bli din egen IT-sjef 8](#_Toc80699010)

[3.4 Må du være en matematikk-kløpper? 9](#_Toc80699011)

[4 Blue-Bot 9](#_Toc80699012)

[4.1 Lag dine egne underlag 10](#_Toc80699013)

[4.2 TacTile Code Reader 10](#_Toc80699014)

[4.3 App 10](#_Toc80699015)

[4.4 Eksempeloppgaver 11](#_Toc80699016)

[5 Code jumper 11](#_Toc80699017)

[5.1 Ulike pod’er, konstanter og variable 11](#_Toc80699018)

[5.2 Lydsekvenser 12](#_Toc80699019)

[5.3 Eksempeloppgaver 12](#_Toc80699020)

[6 iOS Snarveier 13](#_Toc80699021)

[6.1 Slik lager du en snarvei. 13](#_Toc80699022)

[6.2 Eksempler på snarveier 14](#_Toc80699023)

[6.2.1 Ring Oslo taxi 14](#_Toc80699024)

[6.2.2 Nye bøker fra NLB 14](#_Toc80699025)

[6.2.3 Send utklipp 15](#_Toc80699026)

[6.2.4 Les utklippstavle 15](#_Toc80699027)

[6.2.5 Ta opp og send 15](#_Toc80699028)

[6.2.6 Send hvor jeg er på melding 15](#_Toc80699029)

[6.2.7 Tale til engelsk 16](#_Toc80699030)

[6.2.8 Saker fra NRK 16](#_Toc80699031)

[7 Quorum 18](#_Toc80699032)

[7.1 Er Quorum et aktuelt alternativ for grunnskoleelever? 18](#_Toc80699033)

[7.2 Quorum Studio 18](#_Toc80699034)

[7.3 Hvordan ser Quorum-kode ut? 19](#_Toc80699035)

[7.4 Oppgaver 20](#_Toc80699036)

[8 Jaws-scripting 21](#_Toc80699037)

[8.1 Globale og programspesifikke script 21](#_Toc80699038)

[8.2 Forslag til program uten standard Jaws-script 21](#_Toc80699039)

[8.3 Forslag til oppgaver 22](#_Toc80699040)

[8.3.1 Hello world 22](#_Toc80699041)

[8.3.2 Sjekk batteristatus 22](#_Toc80699042)

[8.3.3 Sett inn navn og dagens dato 23](#_Toc80699043)

[8.3.4 Regne ut summen fra 1 til 100 24](#_Toc80699044)

[8.3.5 Kopier aktiv URL (nettadresse) til utklippstavlen 24](#_Toc80699045)

[9 HTML og Javascript 26](#_Toc80699046)

[9.1 Terningkast: Et enkelt eksempel 27](#_Toc80699047)

[9.1.1 Scriptet 28](#_Toc80699048)

[9.1.2 Snik gjerne inn litt kunnskap om universell utforming 28](#_Toc80699049)

[9.2 Oppgaver? 28](#_Toc80699050)

[10 Swift Playgrounds 29](#_Toc80699051)

[10.1 Lag taktile eller godt synlige brett 29](#_Toc80699052)

[10.2 Tastatur 29](#_Toc80699053)

[10.3 Oppgaver 29](#_Toc80699054)

[11 Micro:Bit (Python) 30](#_Toc80699055)

[11.1 Valg av Editor 30](#_Toc80699056)

[11.2 Sette opp VSC for Python og Micro:Bit 31](#_Toc80699057)

[11.3 Oppgaver 31](#_Toc80699058)

[12 Lenker 32](#_Toc80699059)

# Forord

Jeg startet med å kode da jeg var ca. 10 år gammel. Den gangen hadde jeg en liten synsrest, men ble etter hvert blind. Programmering ga meg utløp for skaperglede, kreativitet og mestringsfølelse. Dette er noe seende barn får gjennom tegning, dataspill, fritidsaktiviteter og så videre, men som det kan være krevende for sterkt synshemmede å oppnå. Koding kan være en fantastisk aktivitet for noen, og det er i tillegg et tema alle i grunnskolen skal vite litt om.

En av mine svakheter er at jeg er litt interessert i alt. Det var derfor ganske vanskelig å velge hva jeg skulle studere på universitetet: norsk, historie, biologi, matematikk osb. Det kunne godt ha blitt et av disse fagene, men etter at jeg tok det første informatikk-kurset var det egentlig ingen tvil. Jeg skjønte at dette var noe jeg kunne mestre godt, og ikke minst at det å skape noe passet for meg. Hadde jeg hatt normalt syn hadde jeg kanskje valgt tømreryrket. Det å kode er kanskje ikke så annerledes enn å være tømrer? Tømreren lager hus av ulike materialer. Koderen instruerer datamaskinen til å gjøre ting ved hjelp av digitale byggesaker. Jeg skal ikke tvære ut den sammenlikningen, men det har gitt meg veldig mye nettopp å kunne lage programmer eller forbedre funksjonalitet.

Det å kode profesjonelt er ikke nødvendigvis veldig enkelt, og det er kanskje heller ikke så enkelt å komme i gang med å skrive de første linjene med kode. Dette gjelder alle, men det å være sterkt synshemmet byr på tilleggsutfordringer. Det er utfordringer jeg selv har merket fra dag en. Jeg satt med en kikkertbrille og leste kanskje to, tre tegn av gangen. Da har du liten oversikt, men det går likevel! Ikke bare går det, men det gir kanskje en helt spesiell erfaring. Du må gjøre ting litt annerledes enn de som ser normalt. Den annerledesheten kan være gull verdt tror jeg. Feil; jeg vet det!

I hele livet mitt har jeg måttet være løsningsorientert og søke etter kreative måter å jobbe på eller gjøre private ting jeg har lyst til. Det fører til helt unik kompetanse. Den kompetansen synshemmede barn og unge utvikler bør samfunnet ta til seg! Faktisk mener jeg at mange sterkt synshemmede tenker annerledes og bedre i forbindelse med både koding og andre problemer som skal løses. Derfor oppfordrer jeg alle foreldre, lærere og hjelpere til å la synshemmede unge mennesker få prøve seg, tenke selv og forske på hvordan de kan få til det de vil!

Hvis du bruker skjermleser eller annen hjelpeteknologi kommer du garantert til å få noen utfordringer når du skal starte å kode. Ikke bli skremt av det! Sammenliknet med da jeg startet, fungerer det mye bedre nå! Skjermlesere har for eksempel blitt bedre, og universell utforming er nå også viktig for de som lager kodeverktøy. På verdensbasis er det mange synshemmede programmerere, og det skulle jeg ønske vi også kunne stimulere til i Norge.

Brett opp ermene, kom i gang og sjekk om koding er noe for deg! Verden og Norge trenger mange flere synshemmede kodere!

Morten Tollefsen  
Oslo, august 2021

# Innledning

Da UnIKT lyste ut midler til prosjekter i 2021, og grunnskole skulle prioriteres, tenkte jeg umiddelbart på koding. Koding på ulike måter er viktig i mange situasjoner, både privat og i arbeid. Det er likevel litt andre grunner til at jeg er opptatt av at unge funksjonshemmede får testet ut om de liker å kode. Det har med kreativitet å gjøre, det handler om mestring. Koding kan noen ganger brukes for å kompensere for nedsatt funksjonsevne. Viktigst av alt er at koding kan være skikkelig spennende og morsomt for noen! Sånn var det for meg, og det at koding nå er en langt mer vesentlig aktivitet i skolen gjør det enda viktigere å finne gode løsninger eksempelvis for blinde og svaksynte. Med fagfornyelsen og nye læreplaner innlemmes programmering i flere fag, som matematikk og musikk, med konkrete målformuleringer. Det er mål alle elever skal oppnå, også de som har nedsatt funksjonsevne.

Måten programmering har utviklet seg på, spesielt for begynneropplæring, har endret seg mye de siste åra. Endringene har gjort det mer krevende for sterkt synshemmede, spesielt fordi løsningene som brukes er veldig visuelle og krever dra-og-slipp. Det er heldigvis en økende bevissthet knyttet til universell utforming av programvare. Dette gjelder også kodeverktøy, som blir stadig bedre for stadig flere. Så langt kan det virke som om dette i større grad gjelder profesjonelle verktøy, men vi får tro og håpe at også programvare for koding som brukes i grunnskolen, vil bli mer universelt utformet. Det kan blant annet lovverk være med på å tvinge fram.

Statped har gjort et arbeid knyttet til verktøy som benyttes i grunnskolen. Noen av utfordringene som er identifisert er:

1. Mange verktøy (utstyr, programvare, webløsninger) er uaktuelle fordi det ikke gir noen meningsfull output for blinde (eks. bare blinkende dioder).
2. Noen løsninger kan ikke brukes fordi det er umulig å programmere hvis du benytter skjermleser, kun tastatur eller annen hjelpeteknologi.
3. Noen løsninger kan benyttes, men ofte er det behov for å finne alternative framgangsmåter eller verktøy. Et eksempel er micro:bit. Til denne finnes det 3 offisielle editorer; 1 blokkeditor og 2 teksteditorer. Blokkeditoren kan ikke benyttes med skjermleser. Teksteditorene fungerer heller ikke tilfredsstillende. Det har derimot vist seg at Visual Studio Code er et tilgjengelig alternativ.

Norske myndigheter stiller krav om universell utforming av IKT i utdanningen [1]. Disse kravene skjerpes ytterligere fordi EUs webdirektiv om universell utforming av offentlige nettsteder og mobilapplikasjoner (WAD) blir en del av norsk rett [2]. Vi vet at kompetansen om koding generelt ikke er veldig høy i grunnskolen, og at kunnskap om koding for elever med nedsatt funksjonsevne er nesten ikke-eksisterende. Selv med krav til universell utforming vet vi derfor at det er et stort behov for å gjøre noe på dette feltet. Prosjektet «Koding for alle» har vært tenkt som et lite bidrag, og selv om elever i grunnskolen er en viktig målgruppe for prosjektarbeidet, har vi vurdert løsninger som kan være aktuelle for både yngre og eldre mennesker. Prosjektet må sees på som starten på et arbeid – ikke konklusjonen! Teknologi endres hele tiden, og derfor bør noen følge med på hva som skjer innen koding og universell utforming. Statped har som nevnt sett på de mest brukte kodeverktøyene i skolen. I prosjektet «Koding for alle» har vi derfor valgt å fokusere litt mer på alternative måter å komme i gang på for unge synshemmede. Disse måtene kan også være med på å tilfredsstille kravene i læreplanene.

## Prosjektmål og milepæler

Prosjektets hovedmål var:

Kartlegge og utarbeide løsninger som gjør koding tilgjengelig og spennende for funksjonshemmede, og som tilfredsstiller læreplanene i grunnskolen.

Vår primærmålgruppe var blinde og svaksynte, og følgende milepæler ble definert i prosjektet:

1. Kartlegge eksisterende prosjekter, aktiviteter og funn
2. Gjøre egne vurderinger av aktuelle verktøy
3. Utarbeide tilpasninger
4. Lage en oppgavesamling
5. Testing med synshemmede elever
6. Informasjonstiltak

Dette notatet oppsummerer det vi fant ut i prosjektet. Notatet kan brukes som et utgangspunkt for hvordan det kan være hensiktsmessig å komme i gang med koding. Ingen av oss er helt like, og det som fungerer for noen, fungerer for noen – ikke for alle! Ta derfor en titt på alternativene som beskrives nedenfor og vurder hva du tror vil egne seg for det behovet du kjenner.

## To hensyn å ta

Når synshemmede barn skal starte med koding er det to litt motstridende hensyn å ta:

1. Hvilke verktøy og utstyr bruker andre barn?
2. Hva er mest hensiktsmessig rent pedagogisk?

Kodeverktøy er som oftest ikke universelt utformet [6]. Selv verktøy som fungerer relativt bra, krever mye kunnskap om skjermleser eller skjermforstørrer. Det kan fort bli en minst like stor utfordring å beherske selve verktøyet som å lære koding. Likevel kan det være den beste løsningen at synshemmede gjør tilnærmelsesvis det samme som de med normalt syn.

Det finnes løsninger spesielt utviklet for blinde og sterkt synshemmede. Et slikt verktøy er Code Jumper [21]. Code Jumper, eller andre måter å lære koding på, kan være pedagogisk mer egnet enn standardløsningene som brukes i skolen for noen.

Uansett, standardløsninger som brukes i skolen eller mer spesialiserte kodeløsninger, prøv å få det forståelig og spennende også for unge med nedsatt syn.

## Koding eller programmering?

Jeg bruker begrepene litt om hverandre. I skolen er nok «koding» begrepet som brukes. Jeg er imidlertid så foreldet at for meg heter det «programmering», hehe. I notatet er derfor koding og programmering synonyme begreper.

# Hvorfor skal synshemmede lære koding?

Opplagt svar: Jo, fordi alle skal lære litt om koding i grunnskolen. Et litt mer engasjert svar finner du i Forord. Koding kan være en måte å få utløp for kreativitet, det kan gi mestringsfølelse og ikke minst kan det være skikkelig morsomt. At koding er lærerikt på mange måter er det ingen tvil om. For synshemmede kan det dessverre være mer krevende å komme i gang, men ikke fortvil: Noen av verdens beste kodere er blinde! Akkurat det kan jeg nok ikke bevise, men selv er jeg helt overbevist om det! Det som er helt sikkert er at mange blinde og sterkt svaksynte er profesjonelle kodere! Når de har fått det til, kan selvsagt også andre få det til!

Jeg har noen erfaringer å dele før notatet beskriver noen av de mulighetene som finnes akkurat nå. Koding finnes nesten over alt, og det er godt mulig du vil prøve noe helt annet enn det jeg har tatt med. Supert! Del gjerne erfaringer på Facebook eller andre steder. Men noen forutsetninger og tips gjelder uansett verktøy. Nedenfor finner du mine tips: ikke basert på forskning eller søk på internett, men sammenfallende med det synshemmede forelesere/pannelister fortalte på «National Coding Symposium 2021» [39].

## Kreativitet og løsningsfokus

Hvis du er blind eller sterkt svaksynt, lærer du deg raskt å gjøre ting på alternative måter. Er du forelder eller lærer til en synshemmet jente eller gutt, har du nok også grublet på hvordan ulike ting kan løses. Ta et enkelt eksempel fra hverdagslivet: pusse tennene. Jeg har en egen tannkrem, og tar passe mengde på tunga og overfører så tannkremklatten til børsten. Mye greiere for meg enn å bruke fingeren og prøve å få riktig mengde rett på børsten. Har faktisk aldri spurt andre blinde om hvordan de gjør det, men jeg tipper at veldig mange har funnet samme løsning uten at noen har lært oss det.

Kreativiteten og behovet for løsningsorientert tenking er gull, også når synshemmede skal komme i gang med koding. Mens visuelle måter å kode på er veldig enkelt for barn med normalt syn, må vi som ikke ser finne andre måter å gjøre det på. Vi må skrive tekst, og etter hvert må faktisk alle som skal kode på ordentlig skrive tekst. Kanskje det kan være å jobbe med lyder i stedet for bilder, kanskje kan det være taktil tilbakemelding i stedet for blinkende dioder. Behovene er så ulike at jeg ikke skal komme med spesifikke råd her. Det jeg vil gjøre er å nevne et par eksempel fra min kodeerfaring.

**Fra nesa i skjermen til en mye bedre arbeidsstilling**: Som mange andre svaksynte satt jeg med nesa klint inn i skjermen da jeg fikk min første datamaskin som ni-, tiåring. I tillegg måtte jeg bruke en lupe som riktignok var festet i et brilleglass. Utrolig lite ergonomisk! Ikke var det helt uproblematisk å bevege hodet nok til å se alt på skjermen heller. På skolen brukte jeg en kikkert, og jeg tenkte at det måtte vel finnes kikkerter som kunne festes på briller, og brukes for å se skjermen på lengre avstand. Vi dro til optiker (eller var det øyelege). Han sa at kikkertbriller fantes, men at det ikke ville fungere for meg. Det trodde jeg ikke noe på, dvs. jeg ville bruke den PCen, og da var kikkertbriller en ting jeg bare måtte få sjekket ut. Det ble til at sta-Morten fikk trumfet gjennom ønsket, og en steikdyr kikkerbrille ble laget. Den hadde jeg glede av i mange, mange år! Selv om jeg så veldig lite av gangen, fungerte det himmelvidt mye bedre enn å sitte med nesa i skjermen!

**Dataspill**: Som alle andre startet jeg med å kode helt enkle ting. Etter hvert fikk jeg til mer. Alt før 1980 var det en del som holdt på med dataspill. Det var utrolig få av disse spillene som fungerte for meg. Det eneste var noen tekstbaserte eventyrspill, men jeg ble aldri særlig hekta på det. Strengt talt har jeg aldri vært superinteressert i spill, men i alle fall. Jeg tenkte at det kunne være gøy å lage noen spill med bare lyd. Det gjorde jeg, og litt seinere ble noen av spillene brukt av andre synshemmede (faktisk 15 år seinere, men da selvsagt litt omskrevet). Det er ganske så kult med koding: finnes ikke det du vil ha, så kan du lage det!

Hadde jeg tenkt på alle krumspring og varianter jeg har brukt for å oppnå det jeg vil med teknologi, kunne jeg ha skrevet mange eksempler. Det er imidlertid ikke poenget i dette notatet. Poenget er at du må søke etter alternative måter hvis det trengs. Det å tenke ut løsninger selv gir synshemmede (og de som hjelper synshemmede) mye løsningskompetanse, og det kan brukes også til mange andre problemløsninger!

## Begynn tidlig!

De som er engasjert i et tema som har med barn og unge å gjøre, anbefaler ofte å starte tidlig, så det er ikke akkurat veldig originalt. Men, koding kan være veldig gøy, og det finnes måter å komme i gang på veldig tidlig. Kanskje så tidlig som to, tre, fire år. Barn er så ulike, og det er hverken mulig eller nødvendig å sette en spesifikk alder for når barna kan prøve seg. En måte å starte på, og som er forbausende morsomt for mange barn (jeg har prøvd det ut), er Blue-Bot. Blue-Bot er en liten bille du kan få til å flytte seg med kommandoer (trykk på ulike knapper). Veldig enkelt, så enkelt at du til og med kan simulere det helt uten teknologi.

## Start løpet mot å bli din egen IT-sjef

Det er en stor barriere at skjermlesere og annen hjelpeteknologi er veldig, veldig avansert å bruke. Skjermlesere er tross alt det som er aller mest krevende. Fra et visuelt bilde skal innholdet på skjermen presenteres som tale eller punktskrift. Med tale hører du ett ord av gangen, med en leselist kan du strengt talt ikke lese mer enn et par tegn av gangen. Oversikten blir da liten, og derfor inneholder skjermlesere masse funksjonalitet for å kompensere mest mulig for «det lille vinduet mot verden». Du må lære en hel drøss med hurtigtaster, men jeg kan ikke få sagt det sterkt nok:

**Bruk masse tid på å lære hjelpemiddelteknologien best mulig!**

Det er mange utfordringer knyttet til å lære for eksempel skjermlesere for barn og unge:

* Andre i nærmiljøet (foreldre, søsken, skolekamerater, …) kan ikke teknologien.
* Lærere kan ikke teknologien, i alle fall ikke mer avansert bruk.
* Det gis minimalt med opplæring.
* Den minimale opplæringen som gis i bruk av skjermlesere er gørr kjedelig (i følge flere jeg har snakket med, men vet absolutt ikke om dette gjelder alle).
* …

Finn noe som er gøy for barna, det skal være «leik og lær» ikke «lær og tvær». Hva som er morsomt for barn varierer – like mye blant synshemmede barn som for alle andre barn. Friheten til å gjøre det som er gøy kan være mye mindre for barn med nedsatt syn. Likevel er jeg overbevist om at jeg skulle klart å finne noe som er gøy og teknologibasert for de fleste, og mer spesifikt for å stimulere til læring av hjelpeteknologi.

Datamaskiner er bedre egnet til koding enn mobiler og nettbrett. Det gjelder for så vidt helt generelt, men er enda viktigere for sterkt synshemmede. Skjermlesere for PC (eller Mac) er eksempelvis langt bedre enn på mobiler. Videre er verktøyene som brukes til koding bedre på en datamaskin. Det er altså lurt å komme i gang med datamaskinen og lære mest mulig om bruk av hjelpeteknologien. Alt veldig tidlig må synshemmede (og eventuelle hjelpere) lære å finne løsninger selv. Det er for eksempel nesten ingen som kan bruke en skjermleser i nærmiljøet. Det er noe vi synshemmede er dømt til å leve med. Vi må på en måte bli litt våre egne IT-sjefer! I dag finnes det heldigvis Facebook-grupper og andre sosiale medier der det er mulig å komme i kontakt med andre og få hjelp.

## Må du være en matematikk-kløpper?

Jeg får ofte spørsmål om du må være veldig god i matematikk for å holde på med koding. Mange tror det, men det er helt feil. Selvsagt finnes det koding som krever avansert matematisk kompetanse: kunstig intelligens, maskinlæring, … Mye koding krever derimot ikke spesiell matematisk kompetanse. Derimot tror jeg at koding kan øke forståelsen for logikk og matematikk. Har du et barn som sliter litt med matematikk så ikke tro at det må bety at koding er uaktuelt!

# Blue-Bot

Blue-Bot og Bee-Bot kan brukes alt fra barnehagealder [7]. Bee-Bot er gul og svart og ser ut som ei bie, mens Blue-Bot er gjennomsiktig og lyser. Begge kan programmeres direkte ved hjelp av knappene på ryggen:

* Kjør 15 cm fram
* Kjør 15 cm tilbake
* Snu 90° mot høyre
* Snu 90° mot venstre

Opptil 40 kommandoer kan lagres, så det kan altså kodes ganske avanserte sekvenser. Ofte brukes biene sammen med underlag for å løse oppgaver: «Få bien til å gå til den gule sitronen.», «Lag en sekvens der bien går over alle tallene.» og så videre.

|  |  |
| --- | --- |
| Bee-bot  Bilder: TTS Group, tts-international.com | Et bilde som inneholder kjøkken apparat, kjøkkenmaskin  Automatisk generert beskrivelse |

I tillegg til knappene på ryggen, kan Blue-Bot kodes ved hjelp av en app eller kodebrikker [8]. Appen gir noen ekstra muligheter som repetisjoner og snu 45°. Du kan også få kjøpt brikker som gir de samme mulighetene [9].

## Lag dine egne underlag

For blinde kan det lages taktile underlag. Det kan gjøres ganske enkelt, for eksempel med en papp-plate og tape. Blue-Bot er ikke veldig sterk, og for høye kanter gjør at den spinner eller mister retningen. Eksperimenter gjerne med ulike materialer og taktil merking.

For svaksynte kan det lages underlag med høy kontrast, svart/hvit eller andre visuelle effekter som gjør at det fungerer for den enkelte.

## TacTile Code Reader



Bilde: TTS Group

Med TacTile Code Reader [8] kan du lage en robotsekvens ved hjelp av brikker. En leser har plass til ti brikker, og opp til tre lesere kan kobles sammen i sekvens. Det følger med brikker som tilsvarer knappene på Blue-Bot. Du kan også få kjøpt en utvidelsespakke som gir mulighet for repetisjoner og for å snu roboten 45° [9].

Til tross for navnet, TacTile Code Reader, mangler kodebrikkene taktil merking. Det er imidlertid solide brikker som lett kan merkes med punktskrift eller taktile symboler.

## App

Blue-Bot kan kodes med en app for Android [10] eller iPad [11]. Jeg tenker imidlertid at Blue-Bot er for små barn, og at app-styring har lite for seg. Jeg kan ta feil her, men har i alle fall ikke brukt noe tid på app-styring.

## Eksempeloppgaver

Lag underlag: taktile, med høy kontrast eller liknende.

Det er ganske lett å gi barna oppgaver hvis du først har et egnet underlag:

1. Gå fra «Start» til trekanten.
2. Gå til det øverste høyre hjørnet og få bien til å se rett opp.
3. Finn korteste vei fra «Start» til øverste venstre hjørnet hvis du bare kan gå til tomme ruter.
4. Lag en sekvens der bien passerer alle ruter med en sirkel og så går tilbake til start.
5. …

# Code jumper

Microsoft har utviklet et fysisk kodespråk for barn fra 7 til 11 år som kalles Code Jumper [21]. Ulike «Pod’er» kobles til en «HUB». Output er i form av lyd. Sagt litt mindre teknisk: du kobler noen dingser til en boks og resultatet blir lyder.



Bilde: codejumper.com

## Ulike pod’er, konstanter og variable

Code Jumper består altså av en boks som fungerer som en sentral (HUB) som kobles til PC med Bluetooth. Til sentralen kobler du en pod eller en hel kjede med ulike pod’er. Pod’ene utfører ulike handlinger:

* Spill lyd
* Pause
* Valg («if» i kode-verden)
* Løkker
* Sammenkobling

På podene er det hjul som bestemmer varighet, hastighet og så videre.

I tillegg til pod’er og sentral kommer Code Jumper med konstanter som kan settes i pod’ene, og du har også tilfeldig tall og verdien «uendelig».

## Lydsekvenser

Lydene som spilles av kan være toner, opplest tekst eller annen lyd. Det leveres med en rekke lydsekvenser, og det er ikke vanskelig å lage egne. Denne fleksibiliteten gjør at du kan lage mange morsomme ting ved hjelp av Code Jumper.

## Eksempeloppgaver

Det finnes både oppgaver og opplæringsmateriell på web [22]. Noen oppgaver jeg har brukt:

1. **Spill en melodi**  
   Koble til et lydsett med toner. Lag en sekvens med lyd-pod’er og endre toner og lengde med hjulene på pod’ene.
2. **Lydfyr**Lydfyr er en lydkilde som brukes for å gi synshemmede retningsspill. For eksempel kan et lydfyr brukes når du spiller dart eller ringspill. Bruk Code Jumper for å lage et lydfyr (her får du bruk for en løkke-pod, og «uendelig»-variabelen).
3. **Digital terning**Bruk Code Jumper for å lage en digital terning (du får bruk for tilfeldig tall).
4. **Digital terning 2**  
   Lag en terning som sier det terningen viser, bortsett fra hvis terningen viser 6. Får du 6 spilles det av en fanfare (du får bruk for betingelses-pod’en).
5. …

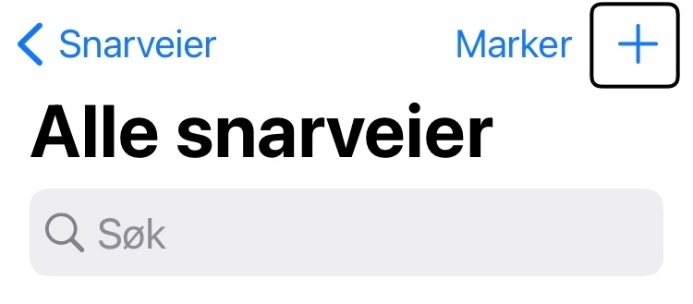
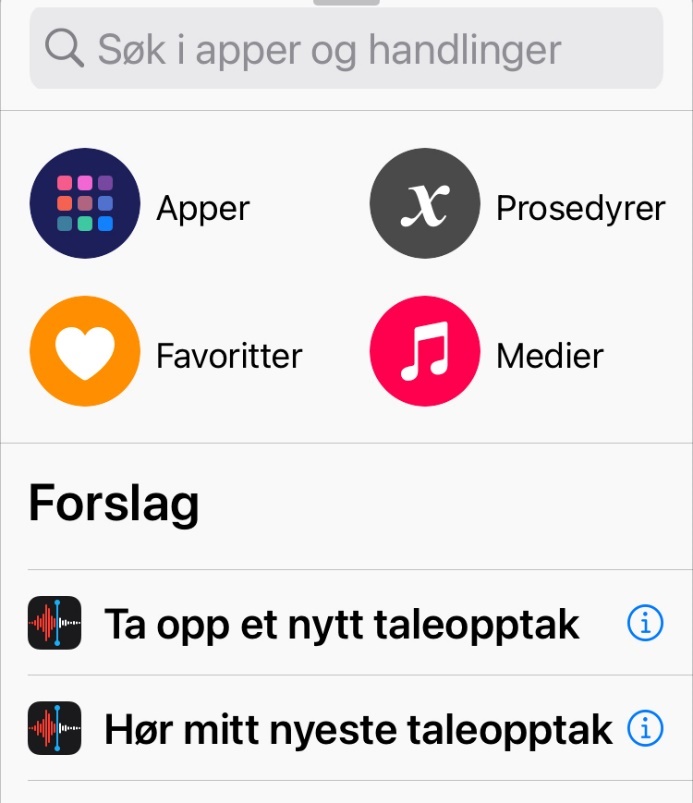
# iOS Snarveier

Innledningsvis skrev jeg (jfr. Start løpet mot å bli din egen IT-sjef) at datamaskin er mer egnet for koding enn mobiler og nettbrett. Det står jeg for, men på iOS-dingser finnes det noe som heter «snarveier». Snarveier minner mye om blokk-programmering. Handlinger legges i en sekvens med enkle muligheter for betingelser, løkker og så videre. Les gjerne brukerveiledningen [16]. Det finnes flere nettsteder der snarveier kan lastes ned, f.eks. Shortcuts Gallery [17]. Slike snarveier kan brukes som utgangspunkt for oppgaver eller tilpasses egne behov. En flott ting med snarveier er at du får tilgang til funksjonaliteten i telefonen eller nettbrettet: kamera, blits, GPS, mikrofon, Bluetooth, nettleser, apper, NFC-leser osv. Snarveier er ganske enkle å forstå, men skal du lage veldig vanskelige ting kan det bli litt uoversiktlig etter hvert.

Jeg har full forståelse for at det å kjøpe en iPhone til en småskoleelev (eller førskoleelev) kan være helt urealistisk for noen. De vil kanskje ikke at barnet skal ha telefon i det hele tatt. For sterkt synshemmede barn derimot tror jeg at mye blir enklere både på skolen og i hverdagen hvis barna lærer teknologi så tidlig som overhodet mulig. Den erfaringen de får med for eksempel VoiceOver, vil være flott også når de skal lage snarveier eller kode på annen måte. En app som BlindSquare kan fortelle deg hvor du går, og du kan sette merker på steder du vil finne fram til. En app som Microsoft AI kan brukes for å fortelle deg hvilken tekst kameraet peker på. Spill, musikk, podcast og underholdning er fint, og alt gjør at du lærer VoiceOver og iOS-dingsen. Mye av det samme gjelder også for Android, men det er rett og slett ikke like bra for oss som er helt blinde. Jeg skal ikke begrunne det her, men jeg mener altså at det er best å satse på iOS hvis du trenger skjermleser.

Snarveier kan kombineres med taleassistenten Siri. Har du laget en snarvei kan du altså bare si navnet på den, og vipps utføres snarveien.

## Slik lager du en snarvei.

1. Åpne Snarveier-appen  
   
2. Velg Opprett snarvei  
   
3. Legg til handlinger (blokker)  
   
4. Velg Neste og gi snarveien et navn.

## Eksempler på snarveier

Nedenfor finner du åtte relativt enkle snarvei-eksempler og hvilke kommandoer du kan bruke. Det er mulig å lage alt fra helt enkle snarveier til ganske sofistikerte greier. Det bør være veldig enkelt å endre litt på snarveiene for å gjøre de riktigere for deg eller bruke eksemplene som oppgaver.

### Ring Oslo taxi

Antall handlinger: 1

Start gjerne med en snarvei med bare en handling. Det kan være fint for å sjekke hvordan det fungerer med Siri etc. Du kan selvsagt bytte ut nummeret og navnet på snarveien for å få en snarvei som er nyttigere for deg.

|  |  |
| --- | --- |
| **Handling** | **Beskrivelse** |
| Ring | Legg inn et nummer og gi deretter snarveien et navn. |

### Nye bøker fra NLB

Antall handlinger: 2

|  |  |
| --- | --- |
| **Handling** | **Beskrivelse** |
| URL | Her legger du inn nettadressen, for eksempel: <https://www.nlb.no/nye-boker> |
| Åpne URL-er | Denne handlingen bruker URL-en (nettadressen) lagt inn i forrige handling og åpner Safari. |

### Send utklipp

Antall handlinger: 2

Dette er den snarveien jeg selv bruker oftest. Den sender meg rett og slett en epost med det jeg har kopiert til utklippstavla. Det kan være tekst fra en nettside, en url, et telefonnummer etc. Jeg bruker den så ofte at snarveien er lagt til som dobbeltklikk-kommandoen bak på iPhone.

|  |  |
| --- | --- |
| **Handling** | **Beskrivelse** |
| Hent utklipp | Handlingen henter det som er på utklippstavla. Innholdet sendes så som en  e-post i neste handling. |
| Send e-post | I handlingen kan du legge til en e-postadresse og eventuelt et emne (hvis du ikke vil legge til emne manuelt når eposten sendes). |

### Les utklippstavle

Antall handlinger: 2

Noen ganger lurer jeg på hva som ligger på utklippstavla. Her er en snarvei som leser det opp.

|  |  |
| --- | --- |
| **Handling** | **Beskrivelse** |
| Hent utklipp | Handlingen henter det som er på utklippstavla. Innholdet leses i neste handling. |
| Les opp tekst | Her leses teksten som ble hentet fra utklippstavla. Du kan velge stemme, hastighet etc. |

### Ta opp og send

Antall handlinger: 4

Denne snarveien starter et opptak (i Snarveier-appen må du tillate at mikrofonen brukes før snarveien får lov til å ta opp). Opptaket sendes som en e-post. Jeg har valgt å sende e-posten til meg selv, helt automatisk, så opptaket sendes umiddelbart med den faste tittelen «Lydopptak».

|  |  |
| --- | --- |
| **Handling** | **Beskrivelse** |
| Ikke forstyrr | Ikke forstyrr skrus på så ikke opptaket blir ødelagt av varsler, noen som ringer etc. |
| Ta opp lyd | Her kan du bestemme opptakskvalitet og når opptaket skal starte og stoppe. |
| Send e-post | Jeg har satt meg selv som mottaker og brukt emnet Lydopptak. Jeg tar bort «Vis skriveark». Det gjør at eposten sendes uten at jeg må legge til noe. Ulempen er at jeg heller ikke blir spurt om mottaker, emne, kopi, blindkopi, eller får mulighet til å skrive noe i e-posten. |
| Ikke forstyrr | Når opptaket er ferdig og epost sendt skrur jeg av «Ikke forstyrr». |

### Send hvor jeg er på melding

Antall hendelser: 3

Denne snarveien laget jeg for lenge siden. Tenkte den kunne være kjekk å ha hvis jeg rotet meg bort og ikke skjønte hvor jeg befant meg. Har heldigvis ikke brukt snarveien mer enn en gang. Da ble jeg satt av i feil sentrumsgate av en taxi. Jeg kunne sende melding til de jeg skulle ha møte med, og de fant meg! Du må gi snarveien tillatelse til å bruke stedstjenester.

|  |  |
| --- | --- |
| **Handling** | **Beskrivelse** |
| Hent nåværende sted | Henter stedet du er på |
| Hent kart-URL | Henter lenke til posisjonen. Den som får meldingen, kan velge lenken og se hvor du er på kartet. |
| Send melding | Her kan du angi mottakere etc. Jeg har ikke lagt inn mottaker, så blir spurt hver gang. |

### Tale til engelsk

Antall handlinger: 9

Denne snarveien lar deg si noe på norsk og oversetter det til engelsk. Den oversatte teksten kopieres til utklippstavla. Kommandoene som slår VoiceOver av/på kan fjernes hvis du ikke bruker VoiceOver. Kanskje du også vil fjerne handlingen som kopierer tekst til utklippstavla. Til slutt i snarveien vises originalteksten og oversettelsen. Velger du OK starter snarveien på nytt, velger du Avbryt avsluttes snarveien.

|  |  |
| --- | --- |
| **Handling** | **Beskrivelse** |
| Angi VoiceOver | Skrur VoiceOver av fordi dikteringen da fungerer litt bedre. |
| Dikter tekst | Lytter til det du sier. Jeg har valgt å stoppe dikteringen når snakkingen slutter (det er en pause). |
| Oversett tekst med Microsoft | Det du har snakket inn oversettes. Her har jeg valgt norsk til engelsk. Du kan selvsagt velge andre språk. |
| Les opp tekst | Den oversatte teksten leses opp. Du kan bestemme språk, hastighet, stemme osv. |
| Kopier til utklipp | Her legges den oversatte teksten på utklippstavla. |
| Tekst | Her settes både norsk og engelsk sammen til en tekst som vises i neste handling. |
| Angi VoiceOver | VoiceOver skrus på igjen. |
| Vis varsel | Norsk og engelsk tekst vises sammen med OK- og Avbryt-knapper. |
| Utfør snarvei | Hvis du velger OK i forrige handling kommer du hit, hvis du velger Avbryt avsluttes «Tale til engelsk» umiddelbart. Snarveien som kjøres i denne handlingen er «Tale til engelsk», altså den samme snarveien som kjøres. |

### Saker fra NRK

Antall handlinger: 16

Dette er en litt mer komplisert snarvei med både løkke, diktering og hvis-handlinger. Snarveien henter de siste 30 nyhetssakene fra NRK. For hver nyhet kan du svare ja hvis du vil ha lest saken, nei hvis den ikke skal leses eller avslutt hvis du ikke vil høre flere saker. Jeg synes det fungerer best hvis VoiceOver skrus av. VoiceOver må skrus på når snarveien er ferdig, eller hvis du velger å avslutte ved å si avslutt.

|  |  |
| --- | --- |
| **Handling** | **Beskrivelse** |
| VoiceOver | Skrus av. |
| Hent objekter fra RSS-strøm | Her velges 30 (eller et mindre/større antall). Adressen til NRKs nyhetsstrøm er: <http://nrk.no/nyheter/siste.rss> |
| Gjenta hver enkelt | Her velges objekter fra RSS-strøm. |
| Tekst | Her lager jeg en tekst som består av dato og tittel. |
| Les opp tekst | Teksten som ble laget i forrige handling leses. |
| Dikter tekst | Her kan du svare Ja, Nei eller Avslutt. Hva skjer hvis du sier noe helt annet? |
| Hvis | Diktert tekst inneholder ja. |
| Tekst | Teksten inneholder Utdrag (en del av objektet som hentes fra RSS-strømmen). |
| Les opp tekst | Utdraget leses |
| Avslutt hvis |  |
| Hvis | Resultatet inneholder avslutt. |
| VoiceOver | Skrus på |
| Avslutt snarvei | Dette avslutter snarveien «tvert». |
| Avslutt hvis |  |
| Avslutt gjentakelse | Her slutter løkka. |
| VoiceOver | Hvis du velger å høre gjennom alle de 30 sakene avsluttes løkka på vanlig måte. Da må VoiceOver skrus på. |

# Quorum

Quorum er et programmeringsspråk og et kodemiljø utviklet spesielt for synshemmede som bruker skjermleser [24, 26]. Quorum kan blant annet brukes for å lage spill, kode Lego-roboter og mye annet.

Det skal være mange skoler for synshemmede som bruker Quorum i undervisningen. Jeg antar at det er riktig, men har ikke gått nærmere i dybden på akkurat det. På National Coding Symposium 2021 [40] ble det opplyst at det er ca. 100 000 som bruker Quorum.

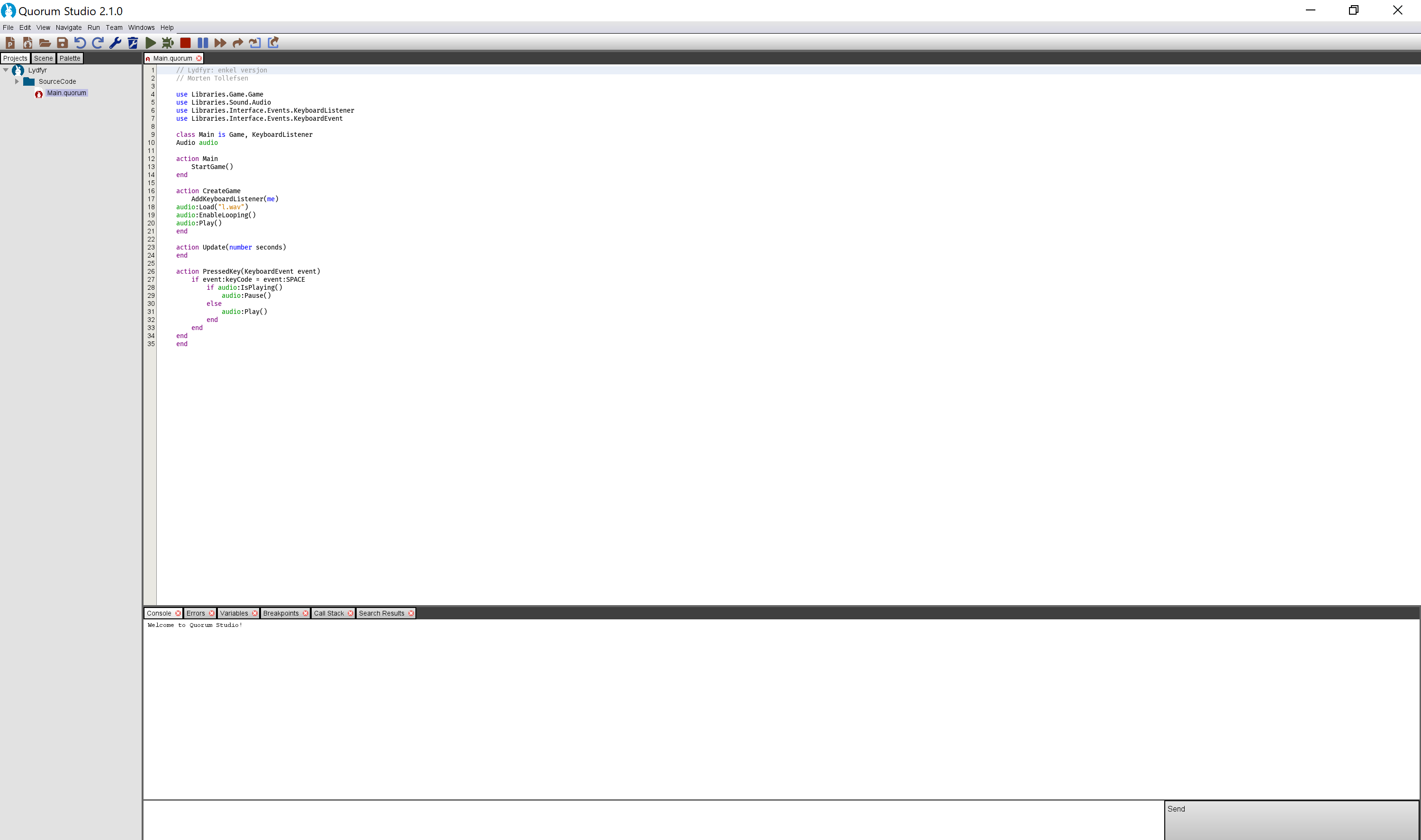
## Er Quorum et aktuelt alternativ for grunnskoleelever?

I Norge har vi ikke spesielle skoler for synshemmede. Quorum kan uten tvil brukes for å lære koding. Utfordringen er at språket ikke har stor utbredelse. Selv om det kan være mer plunder og heft med andre kodeverktøy/språk, tenker jeg derfor at det bør være gode grunner for å velge et språk så få, antakelig ingen, i Norge bruker.

Likevel, jeg må si det: hadde jeg hatt en synshemmet elev som skal lære koding, hadde Quorum sannsynligvis vært det jeg hadde satset på. Med Quorum kan du lage spill, det synes unge folk er gøy. Med Quorum kan du lage apper til mobil, det synes unge folk er gøy. Og viktigst av alt: Quorum gjør det mulig å konsentrere undervisningen om å lære koding, ikke først og fremst slite med plunder og heft og alskens ting som er klønete med skjermleser.

## Quorum Studio

Quorum Studio finnes for PC og Mac. I tillegg kan du jobbe på web [24].



Quorum Studio er det som kalles et Integrated Development Environment (IDE). Enkelt sagt et program som håndterer kodeprosjekter og som har nødvendige verktøy som kodeeditor, debugging-funksjonalitet og så videre.

Det er bare Quorum Studio for PC som har skjermleserstøtte ifølge nettsiden deres [24]. Akkurat dette synes jeg er litt rart når Quorum er laget spesielt for synshemmede.

## Hvordan ser Quorum-kode ut?

Sammenliknet med mange programmeringsspråk er Quorum ganske lettlest. Det er heller ikke massevis av nødvendige, men egentlig unødvendige, tegn (for eksempel for å avslutte hver instruksjon, parenteser for å ramme inn kodeblokker etc).

Nedenfor er det et eksempel på et lite program skrevet i Quorum. Det er rett og slett et «lydfyr», altså en lyd som spilles av for å gi synshemmede en retningsreferanse. Lyden slås av/på med Mellomrom-tasten. Programmet bør være veldig forståelig for de som har kodet noe med klasser, arv og hendelser:

// Lydfyr: enkel versjon

// Morten Tollefsen

use Libraries.Game.Game

use Libraries.Sound.Audio

use Libraries.Interface.Events.KeyboardListener

use Libraries.Interface.Events.KeyboardEvent

class Main is Game, KeyboardListener

Audio audio

action Main

StartGame()

end

action CreateGame

AddKeyboardListener(me)

audio:Load("l.wav")

audio:EnableLooping()

audio:Play()

end

action Update(number seconds)

end

action PressedKey(KeyboardEvent event)

if event:keyCode = event:SPACE

if audio:IsPlaying()

audio:Pause()

else

audio:Play()

end

end

end

end

## Oppgaver

Med Quorum kan du i prinsippet bruke en eksisterende oppgavesamling (laget for et annet språk). På Quorums nettside [24] finnes det gode guider inkludert amerikansk pensum for ulike trinn. Hvordan dette harmoniserer med norsk pensum vet jeg ikke, men jeg antar at det er en ganske stor overlapp. Dersom noen vil gå for Quorum anbefaler jeg å se nøye gjennom oppgaver og guider.

# Jaws-scripting

Jaws-script er ikke nødvendigvis enklere enn Javascript eller andre sammenliknbare språk. Men, det er helt tilgjengelig og hvis du først har Jaws har du alt som trengs ferdig installert. Hovedproblemet med å bruke Jaws er at du kan ødelegge funksjonaliteten til skjermleseren dersom du endrer standardscript/standardfiler. Det er derfor nødvendig å vite hva du gjør og eventuelt ta nødvendige sikkerhetskopier.

Jaws-script er mest aktuelt for de som alt kan en del om skjermleseren. For tenåringer som kan Jaws, vil jeg tro at det kan være både spennende, nyttig og lærerikt å jobbe med Jaws-script.

## Globale og programspesifikke script

Script kan endre hvordan Jaws fungerer. Du kan for eksempel knytte et script til en hurtigtast for å få lest opp noe i skjermbildet, gjøre en sekvens med handlinger (som i en makro) og mye mer. Det er faktisk mulig å gjøre det aller meste ved hjelp av script.

Script kan være globale (fungere uansett hvilket program som er aktivt) eller knyttes til et spesielt program (for eksempel Microsoft Word) eller domene (for eksempel <https://smartja.no/>). Det enkleste er å knytte script til et program eller domene der Jaws ikke har standardscript. Da risikerer du ikke å ødelegge noe.

## Forslag til program uten standard Jaws-script

Som nevnt er det lurt, spesielt i starten, å finne et program som ikke har ferdige Jaws-script. Da risikerer du ikke å ødelegge noe. Det er imidlertid mange av de forhåndsinstallerte Windows-programmene som har ferdige script. I tillegg kan det være lurt å ha en editor/tekstbehandlingsapp for å komme i gang: Da kan du demonstrere script som skriver ut tekst med mer. Jeg foreslår derfor å bruke Klistrelapper-appen (der kan scriptet legge ut tekst i et notat/en klistrelapp).

1. Start Jaws, hvis det ikke er gjort.
2. Åpne Klistrelapper
3. Åpne Scriptverktøy (Jaws+0). Nå har du en tom script-fil.
4. Legg til rette for scriptingen ved å inkludere vanlige konstanter etc. Forslag til innhold:

; Klistrelapper test-fil

include "hjConst.jsh"

include "hjGlobal.jsh"

include "common.jsm"

Nå kan du starte å kode.

Slik blir flyten når du først har begynt scriptet som vist over:

1. Start Klistrelapper og ha Jaws startet.
2. Åpne Scriptverktøy (Jaws+0) og skriv et script.
3. Trykk Ctrl+d hvis du vil legge til en hurtigtast.
4. Trykk Ctrl+s for å lagre og kompilere scriptet.
5. Bytt til Klistrelapper og test om scriptet gjør det du hadde tenkt.
6. Bytt til Scriptverktøy og fortsett kodingen.

## Forslag til oppgaver

Nedenfor har jeg satt opp fem oppgaver. Oppgavene er kun eksempler på hvordan Jaws-script kan brukes for å introdusere ulike kodetemaer.

### Hello world

Et program som skriver ut «Hello World» er gjerne den klassiske starten for å lære et programmeringsspråk. I oppgaven brukes en annen frase, men bruk gjerne «Hello World» eller noe annet hvis du vil.

**Oppgave**: Lag et script som sier «Dette er jammen en fantastisk dag».

**Eksempel på læringsmål**: Åpne scriptverktøy, lage et script, tilordne hurtigtast, kompilere og kjøre scriptet.

**Løsning**:

Script SiNoe ()

SayMessage (OT\_JAWS\_MESSAGE, "Dette er jammen en fantastisk dag!")

EndScript

### Sjekk batteristatus

Jaws har en hurtigtast for å lese batteristatus med mer (Jaws+Shift+b).

**Oppgave**: Lag et script som leser opp antall sekunder batteriet har igjen, eller at dette er ukjent (for eksempel hvis maskinen er koblet til strømnettet).

**Eksempel på læringsmål**: variabler, datatypen integer, konvertere integer til string, if-setning

**Løsning**:

script Batteri ()

var

int b = GetBatteryLifeTimeSecondsRemaining ()

if (b==-1) then

SayMessage (OT\_JAWS\_MESSAGE, "Vet ikke hvor mange sekunder batteri som gjenstår." )

else

SayMessage (OT\_JAWS\_MESSAGE, "Du har " + IntToString (b) + "sekunder batteri igjen.")

endif

EndScript

### Sett inn navn og dagens dato

Denne kan for eksempel legges inn i Word. Hvis du tilordner en hurtigtast til scriptet og trykker tastekombinasjonen i Word, skal Jaws sette inn (endre gjerne navnet, datoen blir selvsagt en annen):

Morten Tollefsen

onsdag 03. mars 2021

**Eksempel på læringsmål**: array, ganske komplisert funksjonskall til TypeString

**Løsning**:

Tips: Lag gjerne litt og litt av TypeString-kallet, og sjekk etter hvert om det blir riktig. Dette er ikke nødvendigvis den smarteste måten å løse oppgaven på, men det finnes ofte mange ulike løsningsforslag.

Script SettInnNavnOgDato ()

Var

StringArray mnd

Let mnd = New StringArray[12]

Let mnd[1] = "januar"

Let mnd[2] = "februar"

Let mnd[3] = "mars"

Let mnd[4] = "april"

Let mnd[5] = "mai"

Let mnd[6] = "juni"

Let mnd[7] = "juli"

Let mnd[8] = "august"

Let mnd[9] = "september"

Let mnd[10] = "oktober"

Let mnd[11] = "november"

Let mnd[12] = "desember"

TypeString ("Morten Tollefsen")

EnterKey (1)

TypeString (GetDayOfWeek (SysGetDate ("yyyy/MM/dd"), "L") + " " + SysGetDate ("dd") + ". " + mnd[SysGetDate ("MM")] + " " + SysGetDate ("yyyy"))

EndScript

### Regne ut summen fra 1 til 100

Lag et script som regner ut summen fra 1 til 100.

**Eksempel på læringsmål**: løkke

**Løsning**:

Script Sum1til100 ()

var

int i,

int x

let i = 0

For x = 1 to 100

i = i + x

EndFor

SayMessage (OT\_JAWS\_MESSAGE, "Summen er " + IntToString (i))

EndScript

### Kopier aktiv URL (nettadresse) til utklippstavlen

Åpne nettleseren. Lag et script som kopierer nettadressen du står på til utklippstavla.

**Eksempel på læringsmål**: Bruke hurtigtaster for å navigere i nettleseren og finne riktige funksjoner. Nøyaktig hvilke tastetrykk som skal brukes kan variere mellom nettlesere. Jeg brukte Edge da scriptet ble laget.

**Løsning**:

Script mtURL2Clipboard ()

var

string aURL

TypeKey ("Alt+d")

Delay (2)

aURL=GetSelectedText ()

CopyToClipboard (aURL)

TypeKey ("f6")

EndScript

# HTML og Javascript

Nettsider er nesten alltid laget med en kombinasjon av HTML, stilark (CSS) og Javascript. I tillegg brukes selvsagt mye annen teknologi. Det å lære HTML vil kunne være nyttig for veldig mange! Selv om du kan publisere ting på web uten å kunne noe som helst om HTML, er det en stor fordel å kunne litt om dette «språket».

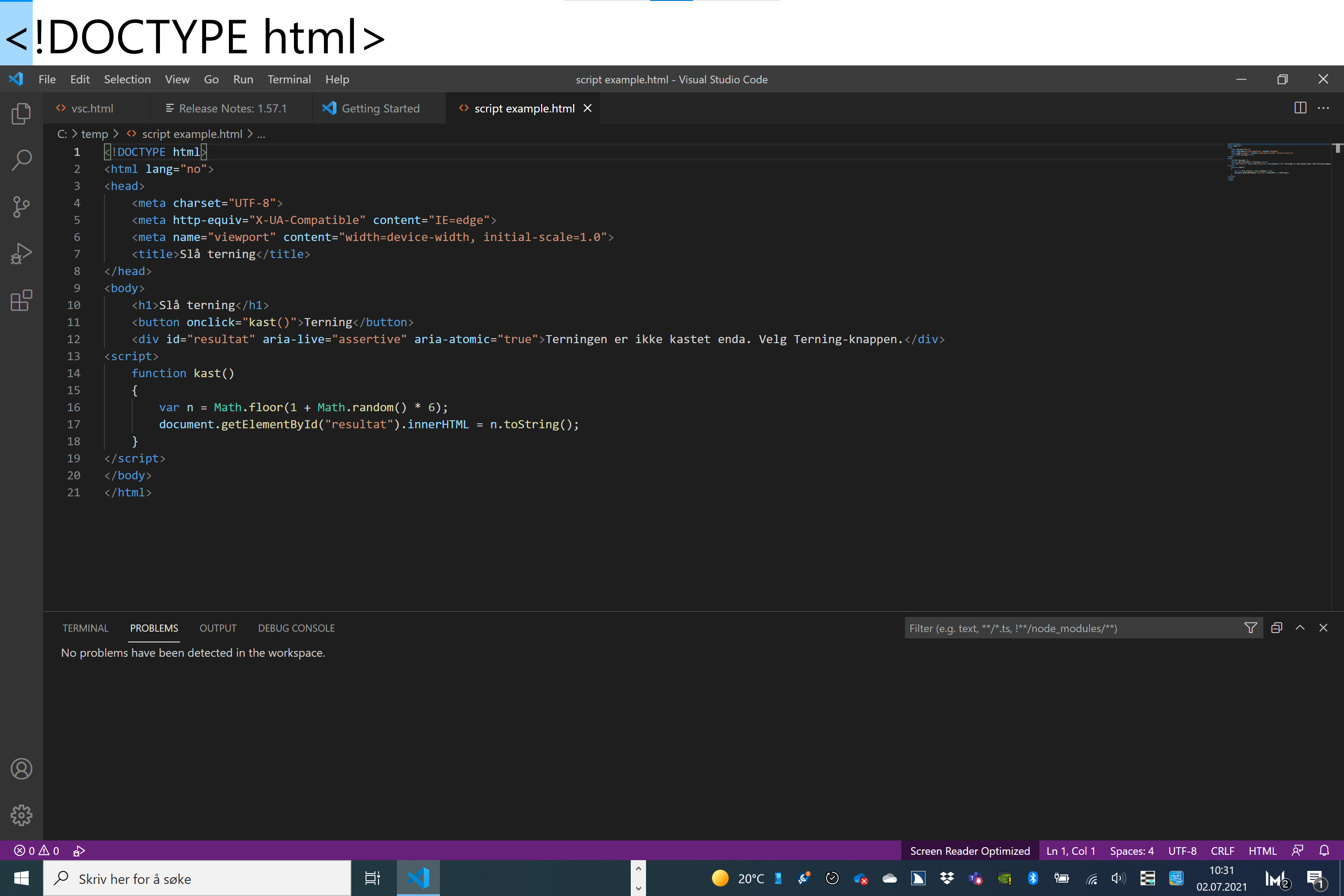
Litt forenklet kan vi si at:

* HTML brukes for å vise tekst, bilder, lenker – altså innhold.
* CSS brukes for å angi utseende (farger, plassering, størrelse, …)
* Javascript brukes for å få ting til å skje.

Egentlig er jeg overrasket over at ikke HTML og Javascript er mer brukt til å komme i gang med koding. HTML er veldig godt egnet for de som bruker skjermleser. Det skyldes at alt «pakkes inn» i tag’er. Skal du ha en overskrift på nivå én, er for eksempel koden:

<h1>Dette er en overskrift</h1>

Mange vil nok si at HTML ikke er koding, og det kan jeg være enig i. Likevel har det mye til felles, og HTML kan kombineres med Javascript (eller andre script-språk).



Jeg bruker Visual Studio Code for å skrive HTML og Javascript. Det er imidlertid helt OK å bruke Notisblokk, Word eller et annet program som kan lagre tekst.

Det finnes mange lærebøker, gratiskurs og andre ressurser på web om HTML, CSS og Javascript. For blinde er det kanskje ikke så enkelt å jobbe med stilark, men det er ikke bortkastet å vite litt om utseende selv om du ikke ser! Jobber du profesjonelt med webutvikling må du selvsagt også kunne litt CSS, men det er helt unødvendig for å starte med koding!

## Terningkast: Et enkelt eksempel

Slik kan koden til en HTML-side se ut:

<!DOCTYPE html>

<html lang="no">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Slå terning</title>

</head>

<body>

<h1>Slå terning</h1>

<button onclick="kast()">Terning</button>

<div id="resultat" aria-live="assertive" aria-atomic="true">Terningen er ikke kastet enda. Velg Terning-knappen.</div>

<script>

function kast()

{

var n = Math.floor(1 + Math.random() \* 6);

document.getElementById("resultat").innerHTML = n.toString();

}

</script>

</body>

</html>

Det aller meste i koden er ting som alltid inngår i en HTML-fil. Det egentlige innholdet begynner rett etter body-tag’en. Der er det en overskrift og en tekst som sier: Terningen er ikke kastet enda. Velg Terning-knappen.

### Scriptet

Nederst i HTML-fila ligger det et script med funksjonen kast. Denne funksjonen utføres når du velger Terning-knappen. Funksjonen trekker et tilfeldig tall fra 1 til 6 og legger tallet på den plassen der det tidligere sto at terningen ikke var kastet.

### Snik gjerne inn litt kunnskap om universell utforming

Hvis du bruker skjermleser og velger Terning-knappen vil fokus være på knappen, og egentlig vet du ikke at noe har skjedd. Du må flytte leselist eller tale dit endringen har skjedd (akkurat i dette tilfelle rett under knappen). Ved å bruke en standard som heter WAI-ARIA, kan du imidlertid si i fra at det som skjer på et gitt sted i koden skal leses opp av skjermlesere. Derfor har jeg lagt til attributtene aria-live og aria-atomic på div-tag’en, der terningkastene vises.

Etter hvert som kunnskapen om HTML og Javascript øker, kan mange ting som har med universell utforming å gjøre inngå i koding og eventuelle oppgaver elevene får. Og jeg synes alle godt kunne ha dette som pensum, uansett nedsatt syn eller ikke.

## Oppgaver?

Det burde være enkelt å gi oppgaver der du bruker HTML og Javascript. Litt flåsete sagt: Du kan bruke hvilke som helst oppgaver og ev. skrive de litt om. Unntaket er kanskje hvis du skal styre roboter eller liknende.

# Swift Playgrounds

Swift Playgrounds finnes for Mac og iPad. Dette skriver Apple [12]:

Swift Playgrounds er en banebrytende iPad-app hvor du lærer Swift på en interaktiv og morsom måte. Du trenger ingen forkunnskaper, så appen er perfekt for nybegynnere. Ved å løse oppgaver lærer du de grunnleggende prinsippene i Swift – et kraftig programmeringsspråk utviklet av Apple, som brukes til å lage mange populære apper. Etter hvert får du nye utfordringer, og kan boltre deg i mer avanserte lekeplasser designet av Apple og andre bransjeledende utviklere.

Dette høres fint ut, og appen er universelt utformet. Litt mer presist kan vi si at den lar seg bruke med hjelpeteknologi som VoiceOver eller Zoom:

Det er motiverende å lære programmering med Swift Playgrounds. Appen leveres med et komplett sett Apple-utviklede leksjoner. Spill deg gjennom grunnprinsippene i Fundamentals of Swift, hvor du bruker ekte kode til å styre en figur gjennom en 3D-verden. Deretter går du videre med mer avansert koding.

Min erfaring er at det som er vanskeligst er å skjønne oppgavene, dvs. å klare å forstå hvordan den virtuelle verden ser ut, og å avgjøre hvordan Byte (figuren du styrer) skal kodes. Etter hvert som leksjonene blir vanskeligere, blir det faktisk enklere å skjønne hvis du bruker VoiceOver.

## Lag taktile eller godt synlige brett

Jeg tror Swift Playgrounds hadde vært mer anvendelig for sterkt synshemmede dersom det var laget taktile versjoner av brettene der Byte skal bevege seg. Jeg vet at noen har laget slike brett, men er usikker på om de kan bestilles (fant det ikke da jeg lette på web).

Det finnes podkaster med beskrivelse av hvordan Swift Playgrounds fungerer med VoiceOver: [14, 1].

## Tastatur

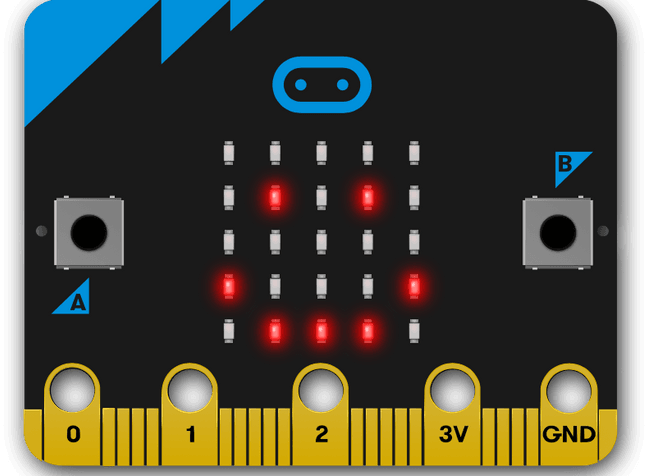
Spesielt hvis du bruker VoiceOver, men sannsynligvis for alle, er det enklere å skrive på et fysisk tastatur enn på skjermtastaturet til iPad. Koble derfor til et fysisk tastatur.

## Oppgaver

Siden Swift Playgrounds er en oppgavesamling har jeg ikke laget spesielle forslag i dette notatet.

# Micro:Bit (Python)

BBC Micro:Bit er en liten, billig datamaskin som kan brukes for å lære koding. Se [1] for en kort introduksjon. Micro:Bit kan kobles til en drøss med sensorer og annet tilleggsutstyr, og maskinen er fin for å lære koding. I utgangspunktet har Micro:Bit bare en matrise med 9 x 9 led-lys for output, men det finnes små høyttalere som kan tilkobles. Høyttaleren kan spille av syntetisk tale og annen lyd.



Bilde: microbit.org

Python brukes mye som introduksjonsspråk til koding, og er det vanligste språket også for å kode Micro:Bit. Det finnes mange ressurser på web for å lære Python [30, …].

## Valg av Editor

For synshemmede er MakeCode og Visual Studio Code de mest aktuelle editorene for koding av Micro:Bit. MakeCode kan brukes både til blokkprogrammering og tekst-koding. For sterkt synshemmede er koding med tekst enklest å få til, dvs. i praksis det eneste alternativet. Riktignok jobbes det med løsninger for blokkprogrammering, men foreløpig ingen løsninger som kan tas i bruk, så langt jeg kjenner til. iOS-snarveier er for så vidt blokkprogrammering, men det er ikke aktuelt med Micro:Bit.

Statped anbefaler Visual Studio Code (VSC) [31], og den vurderingen er jeg enig i:

* VSC har større tilpasningsmuligheter og flere hurtigtaster.
* VSC er et veldig mye brukt verktøy som elevene kan ha stor nytte av til mer enn Python-koding.

For å få VSC til å fungere med Python og Micro:Bit, er det litt styr for å komme i gang (i alle fall hvis du ikke alt er en dreven VSC-bruker). Jeg tar med installasjonen i neste underkapittel, og for å unngå misforståelser, er framgangsmåten nokså identisk med den som finnes i Statpeds veiledning [31]. Bare for ordens skyld: Dette kan settes opp på alternative måter, litt avhengig av preferanser og erfaring.

## Sette opp VSC for Python og Micro:Bit

Du må laste ned Python, VSC og noen utvidelser som gjør at VSC og Python kobles sammen.

**Python**:

1. Installer Python fra:  
   <https://python.org/downloads/>
2. Kryss av for «Add Python to path» og velg «Install now».

**Visual Studio Code (VSC)**:

1. Installer Visual Studio Code fra:  
   <https://code.visualstudio.com/>
2. Bruk standardvalgene i installasjonen.
3. Åpne VSC.
4. Åpne utforskerruten med (Ctrl+Shift+e) og velg en mappe for python-prosjektene.

**Utvidelser**:

1. Åpne utvidelsesruten: Ctrl+Shift+x
2. Installer utvidelsene «Python fra Microsoft» og «Micro:Bit fra Statped».
3. Start Visual Studio Code på nytt.
4. Åpne Command Palette (Ctrl+Shift+P).
5. Velg kommandoen micro:bit Prepare og vent til du har fått beskjed om at det var vellykket.
6. Trykk Ctrl+Shift+P og velg kommandoen Reload Window.

## Oppgaver

Statped har laget fine oppgavesamlinger [33, 32], og veldig mange av oppgavene som finnes på web kan brukes eller tilpasses synshemmede. Bruk for eksempel høyttaler til output i stedet for led-lysene. Jeg har ikke tatt med oppgave-forslag til Micro:Bit/Python, siden det alt finnes oppgavesamlinger som kan kompletteres med Statpeds oppgaveforslag.

# Lenker

1. **Innfører krav om universell utforming av IKT i utdanningen**  
   <https://www.regjeringen.no/no/aktuelt/innforer-krav-om-universell-utforming-av-ikt-i-utdanningen/id2521801/>
2. **EUs webdirektiv blir en del av norsk regelverk**  
   <https://uu.difi.no/nyhet/2018/09/eus-webdirektiv-blir-en-del-av-norsk-regelverk>
3. **Programmering i skolen**  
   <https://www.udir.no/globalassets/filer/programmering_i_skolen.pdf>
4. **Kodeverktøy - tilgjengelighet for elever med synshemning**  
   <https://www.statped.no/laringsressurs/syn/kodeverktoy-og-tilgjengelighet/>
5. **Everyone can code**  
   <https://www.smartja.no/everyone-can-code/>
6. **Can Everyone Code? Promising Tools for Coding with Accessibility for All**  
   <https://www.closingthegap.com/can-everyone-code-promising-tools-for-coding-with-accessibility-for-all/>
7. **Koding med Blue-Bot**  
   <https://www.kidsakoder.no/2016/08/21/koding-blue-bot/>
8. **TacTile Code Reader**  
   <https://www.tts-international.com/tactile-code-reader/1010494.html>
9. **TacTile Code Reader Tiles Extension Pack**  
   <https://www.tts-international.com/tactile-code-reader-tiles-extension-pack/1009828.html>
10. **Blue-Bot (Google Play)**  
    <https://play.google.com/store/apps/details?id=tts.bluebot&hl=en_US&gl=US>
11. **Blue-Bot (iPad/App Store)**  
    <https://apps.apple.com/app/blue-bot/id957753068?ign-mpt=uo%3D4>
12. **Swift Playgrounds**  
    <https://www.apple.com/no/swift/playgrounds/>
13. **Teaching Code (Apple)**  
    <https://www.apple.com/education/k12/teaching-code/#accessibility>
14. **Applevis (podcast): Learn to Code with Swift Playgrounds on iPad**  
    <https://www.applevis.com/podcasts/learn-code-swift-playgrounds-ipad-part-1>
15. **Demo of Lego EV3 robot with Swift Playground and the VoiceOver screen reader**  
    <https://www.youtube.com/watch?v=yDUHNxugu6s>
16. **Brukerveiledning for snarveier**  
    <https://support.apple.com/no-no/guide/shortcuts/welcome/ios>
17. **Nedlastbare snarveier**  
    <https://shortcutsgallery.com/>
18. **Basics of scripting (Jaws)**  
    <https://support.freedomscientific.com/Content/Documents/Other/ScriptManual/01-0_Introduction.htm>
19. **Enhancements in Scripting (Jaws)**  
    <https://support.freedomscientific.com/support/jawsdocumentation/ScriptingInfo>
20. **Microsoft vil lære blinde barn å kode**  
    <https://www.smartja.no/microsoft-vil-laere-blinde-barn-a-kode/>
21. **Code Jumper**  
    <https://codejumper.com/>
22. **Code Jumper Resources**  
    <https://codejumper.com/resources.html>
23. **Code Jumper-an inclusive physical coding language**  
    <https://education.microsoft.com/en-us/course/b38ae828/overview>
24. **The Quorum Programming Language**  
    <https://quorumlanguage.com/>
25. **Quorum Reference Pages**  
    <https://quorumlanguage.com/reference.html>
26. **Quorum: An Accessible Programming Language**  
    <https://www.youtube.com/watch?v=X29BuzGHlBs>
27. **Introducing the BBC micro:bit - BBC Make It Digital**  
    https://www.youtube.com/watch?v=Wuza5WXiMkc&feature=emb\_imp\_woyt
28. **Visual Studio Code og Python for synshemmede**  
    <https://github.com/oivron/coding-without-seeing/wiki/Visual-Studio-Code>
29. **Get Started Tutorial for Python in Visual Studio Code**  
    <https://code.visualstudio.com/docs/python/python-tutorial>
30. **BeginnersGuide/Programmers (Python Wiki)**  
    https://wiki.python.org/moin/BeginnersGuide/Programmers
31. **micro:bit med Python: Komme i gang**  
    <https://github.com/oivron/komme-i-gang-microbit/wiki/micro:bit-med-Python:-Komme-i-gang>
32. **Komme i gang med micro:bit**  
    https://github.com/oivron/komme-i-gang-microbit
33. **super:bit-oppdraget**  
    <https://github.com/oivron/komme-i-gang-microbit/blob/master/superbit-oppdraget.md>
34. **Micro:bit Educational Foundation**  
    <https://microbit.org/>
35. **BBC micro:bit for Visually Impaired Learners**  
    <http://physicalcomputing.co.uk/2018/01/03/bbc-microbit-for-visually-impaired-learners/>
36. **Microsoft MakeCode**  
    <https://www.microsoft.com/en-us/makecode>
37. **Microsoft MakeCode for micro:bit**  
    <https://makecode.microbit.org/>
38. **Accessibility in Visual Studio Code**  
    <https://code.visualstudio.com/docs/editor/accessibility>
39. **2021 National Coding Symposium – Agenda**  
    <https://aphconnectcenter.org/coding/agenda/>
40. **Coding Symposium: Day 1**  
    https://www.youtube.com/watch?v=sRzQCu9JwuQ