Elm

Elm-kurset ble startet i 2017, og er under utvikling. Oppgavene dekker vil etter runde 2 i 2017 være nok til at man kan være med på to runder Elm uten å kjede seg.

Jeg (Teodor) ville starte med Elm kurset etter å ha deltatt på Web-kurset. Webteknologi er fryktelig komplisert. Mange, mange teknologier flettet sammen. Så skal man skrive JavaScript. På en god måte. For å lage noe som teller et tall, er det ikke helt trivielt (for barn og voksne) å lage filene man trenger, linke de sammen på rett måte, og skrive JavaScript som henter ut data, behandler den, og dytter data tilbake. Så skal dette skaleres av en 12-åring. Ouch. Elm har noen fine tanker om dette:

a. Gjør alt i en Elm-fil. Ikke ørten filer åpne i en teksteditor av et barn som ikke har brukt en teksteditor før kurset, og kommer fra Scratch, hvor ting ikke engang må lagres.
b. De fleste feilene fanges av kompilatoren, som gir nyttige tips til hva som kan være feil. Ikke “undefined is not a function”.

Nerdesnacks: Elm er et elegant rent funksjonelt programmeringsspråk med arv fra Haskell, ML og F#. Dette gjør at vi kan skrive kode som ikke krasjer i runtime. Med fin syntax og godt typesystem trenger vi heller ikke skrive så himla mye kode for å få noe til å skje.

Jeg vil hjelpe til!

Kult! Vi er åpne for nye veiledere, på alle nivåer.

Jeg har ikke brukt Elm før!
Prøv å jobbe deg gjennom oppgavene for dag én og dag to. Slå opp i guiden (anbefalt) hvis du lurer på noe. Ikke vær redd for at dette er fryktelig vanskelig.

Jeg har noe erfaring med Elm
Sjekk gjerne ut oppgavene vi skal jobbe med på dagen du er med. Vi følger planen under. Hør eventuelt med Teodor.

Jeg vil skrive oppgave!
Herlig! Erik og Perry har laget oppgaver så langt.

- Lag Pull Request som vanlig
- Teodor kan bistå med det tekniske og/eller gi tilbakemelding på oppgaven.

Kan jeg bidra med noe annet?
Næh, så fint at du spør! Du har et meget godt utgangspunkt for å evaluere oppgaver, nettopp fordi du ikke allerede kan alt oppgaven skal lære bort.

- Noen av oppgavene har veldig mye tekst, og få bilder. Det er alltid bra at barna ser hva de driver med. Gjør oppgaven selv og ta bilder underveis. Se forrige avsnitt for å få lagt inn hva du har gjort.
- Ingen av oppgavene har så langt lærerveiledning.

Slik fungerer en typisk dag

Vi baserer oss tungt på å ha gode oppgaver. Dette lar barna jobbe i eget tempo, og gir en god referanse de kan se tilbake på når de ikke husker hvordan noe ble gjort.

På en typisk dag jobber vi med en eller to oppgaver. Tidsbruken ser da omtrentlig slik ut:

1. Jeg viser konseptet vi skal lære på projektor, og viser oppgaven vi skal jobbe med (5-10 min).
2. Barna jobber med oppgave. Underveis går vi rundt og ser hvor langt folk kommer. Hvis mange blir ferdige, går vi videre til ny oppgave. (Vise konsept, vise oppgave, jobbe med ny oppgave.)

Progresjon dag-til-dag

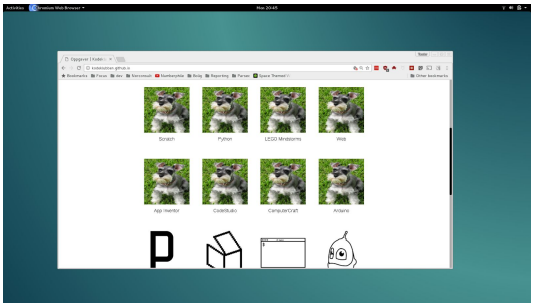
Det er i dag *ett* Elm-kurs. Vi skiller ikke mellom videregående Elm og grunnleggende Elm. Det betyr at barna følger "sin egen" progresjon. Det er derfor flere enn fem dager. De som kommer tilbake på kurset fortsetter da på dag 6.

Referanser for veiledere:

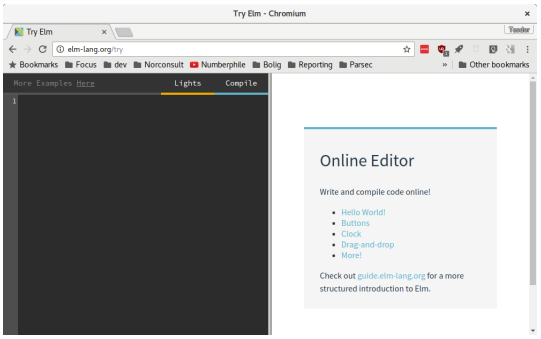
- Den offisielle guiden er knallbra.
- Introduksjonsvideo til Elm for Javascript-utviklere
- Oppgaver linket inn under.

Kurset følger spillelisten *Elm fra hei til spill*. Vi gjør 1-2 oppgaver per dag.

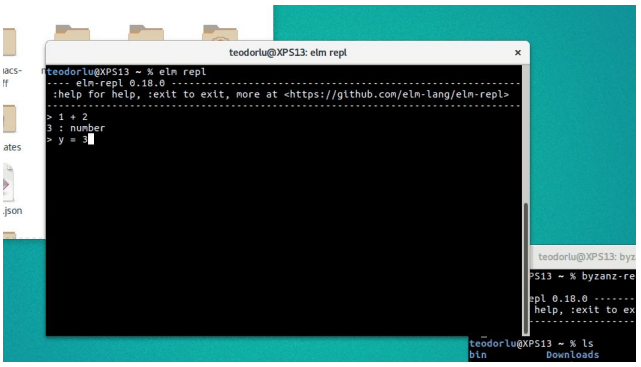
1. **Intro til Elm og HTML**. Hva er HTML? Hva er Elm? Hvordan ser Elm typisk ut?
    a. **Vis grundig hvor oppgavene ligger**. Hvordan finner vi oppgavene? bit.ly/kodeklubben-elm linker til Teodors oppgave-fork. Vis også hvordan man finner HTML-introduksjonen (Ut, annen kategori, inn i Web-kurset, velg *Hvor er HTML? Jeg ser den ikke!*).
    b. **Leke med HTML**. Formålet her er å gi en forståelse av HTML som en trestruktur; at ting kan være inne i andre ting. Dette legger bakteppet for hvordan HTML fungerer. Oppgaven er mest lek, sitte i utviklerverktøyene som følger med Chrome og leke seg med en eksisterende nettside



c. **Leke seg med Try Elm**. Trenger ikke installere noe første dag. Gå rundt i eksemplene til høyre, prøv å endre ting, og se hva som skjer.

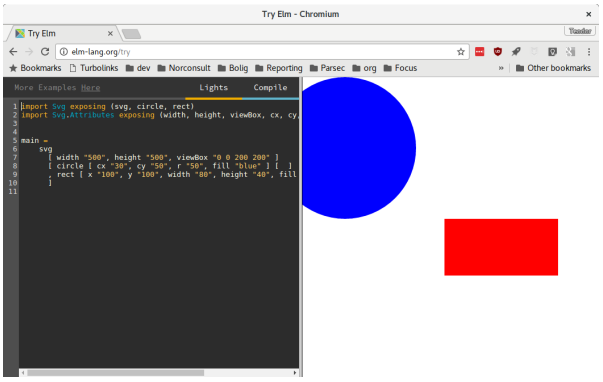


2. **Elm-plattformen, funksjoner og moduler**. Hva trenger vi av verktøy for å bruke Elm? Hvordan splitter vi koden vår i funksjoner og moduler?
    a. **Forklare hvordan toolingen til Elm fungerer**. Vi laster ned Elm plattform, som inneholder flere programmer. Vi bruker to av dem: Elm Reactor bygger automatisk siste lagrede versjon av en nettside, og Elm Repl lar oss definere og teste funksjoner interaktivt.
    b. **Jobbe med oppgaven Elm repl og funksjoner**. Oppgaven guider også til hvordan vi setter opp verktøyene til å fungere lokalt. Ved seinere oppgaver kan barna velge selv om de vil jobbe i Try Elm (enkel skriv Elm og se hva som skjer), Ellie (mer avansert Try Elm) eller med lokale filer.



    c. **Jobbe med oppgaven Elm Reactor og moduler**. Oppgaven viser hvordan vi kan splitte opp Elm-programmer i flere filer. Dette gjør vi med moduler; en fil er en modul.
3. **Input fra brukeren**. Måten Elm fungerer på gjør at vi må *abonnere* på hendelsene vi er avhengig av. Brukerinput er en type hendelse.
    a. **Vise eksempel fra Try Elm**. Eksempelet er et inputfelt som reverserer tekst. Kan vi legge til “Reversert: ” foran teksten? Utropstegn etter?
    b. **Gjøre input-oppgave**. Oppgaven likner eksempelet.
        i. Forbedringspotensiale: flere bilder!
4. **Tegning med SVG**. Vi viser hvordan vi kan bruke SVG til å tegne figurer, og ser hvordan Stefan Kreitmayer har brukt SVG til å lage spill. Vi spiller litt, har det gøy, og leker med fysikken i spillet.
    a. **Demonstrerer SVG**, hvordan vi får med pakkene vi trenger, hvordan tegningen fungerer, hvordan SVG fungerer sammen med HTML; at du kan putte SVG-elementer inne i HTML-elementer.

b. **Gjøre SVG-oppgave**. Tegne firkanter, sirkler og streker. Bruke SVG-dokumentasjonen til Elm og Mozilla. Koordinatsystemet. Bezierkurver og kontrollpunkter. Ellipser. Bruke SVG inne i HTML.



c. **Leke med Elm Joust**, et spill i nettleseren for to personer skrevet i Elm. Lære å navigere i andres kode, bygge spill i terminalen, endre fysikken i spillet, endre fargene i spillet, justere hastighet og kollisjoner. Denne oppgaven var populær!

5. **Tid i Elm**. Vi trener på hvordan vi kan gjøre oss avhengig av tid i Elm.
   a. **Vise tid med klokke-eksempelet**. Hvordan går sekundviseren? Hvordan kan vi få den til å gå fortere?
   b. **Gjøre Tell sekunder-oppgaven**. Hva er et abonnement (subscription)? Viser relativt mye kode samtidig
6. **Kontrollere programmet med musen**. Vi går videre fra Html.beginnerProgram til Html.program, som lar oss reagere på andre ting enn hva som skrives inn i inputfelter. Vi bruker så musen til å styre en sirkel vi tegner i SVG
7. **Programvareutvikling**. Vi går tilbake til begynnelsen og ser på hvordan vi kan bygge opp programmer med funksjonskomposisjon. Oppgaven gir trening i å skrive funksjoner ved å tenke selv.
8. **Fraktaler 1**. Vi begynner på Fraktaler-oppgaven. Erfaringer videre: vanskelig oppgave med stort sprang i ferdighet fra forrige oppgave. Ikke anbefalt.
9. **Fraktaler 2**. Vi fortsetter på Fraktaler-oppgaven.
10. **Spill!** Liten introduksjon til hvordan vi kan få noe til å bevege seg på skjermen.