IT1 Teori

Kap 1:

HTML er et markeringsspråk og brukes når i koder nettsider. Nettsidene vises i egne programmer som heter nettleser. Vi kaller taggene og innholdet vi skriver for å lage en nettside for HTML-kode. Mellom taggene head /head skriver vi inn kode for det som kalles metadata. Metadata er data som ikke blir synlig på nettsiden, i title hvis du skriver det kommer det opp øverst for nettleseren din. Mellom taggene body /body skriver vi inn koden for innholdet som ble synlig på nettsiden. HTML element består som regel av starttagg, innhold og en sluttagg. Vi legger ofte HTML elementer inn i andre HTML-elementer. Dette er nøstede HTML elementer. Lenker mellom nettsider kaller vi interne lenker. Lenker til nettsider på andre nettsteder kaller vi eksterne lenker. Vi setter inn et skjema med form-elementet, der vi kan ha forskjellige ting i et skjema. Om det gjelder et innskrivningsfelt der vi bruker input, her kan vi ha forskjellige typer, slik som text, number, email etc. Enn knapp som vi lager er jo et typisk button element. Når brukeren bruker knappen, så vil form action sende dette til det som er angitt i form. Hvis du ønsker å spare på det, kan du bruke da javascript. Assigner id til skjemaet, setter det i form action feltet, deretter kan du i javascript filen be den skrive infoen til en txt fil. En nedtrekks liste er typisk select elementer der option er valgene i nedtrekks listen. Required elementer er det vi mener er påbudt i et skjema, her kan vi også sette en placeholder for å ha en eksempelverdi slik at brukeren skjønner hva som skal føres inn.

Tagger:

Width setter bredde

Height setter høyde

Audio setter lyd, mp3 fil

A href setter link til nettsted, en klikkbar lenke.

Table er for tabeller

Tr er tabell row

Thead er tabell head

Tbody er tabell body

Td er table data

Th er table head

Form er skjema

Form action = «» hvor vi vil sende dataen

Select lager nedtrekksliste

Option = valgene i nedtrekklisten, typisk «Mann, dame»

Radio er typisk «en mulighets svar»

Label er bare en navnsettings metode

Div er diverse klasser, her kan vi si div class = «form-group» for da å danne et skjema som er en del av den klassen.

Input, innskrivningsfelt. Typer : «text», «number», «date», “color”, “range = glider”, “password”, “checkbox”

Kap 2:

CSS er det som brukes til å endre utseendet til en nettside. Her er det forskjellige måter du kan endre utseendet. Du kan skrive CSS kode direkte inn i headeren ved bruke av style /style, der vi definerer de forskjellige taggsene med forskjellige layouts. Da kan vi si at body skal ha en background color på sort, som vi kan representere med farge navn, #000000 indeksering, eller rgb. # indeksering er at vi bruker hexidecimale verdier for å definere en farge. Der verdiene er basert på en skala fra 000000 til ffffff. Rgb er å definere farger etter intensiteten til rød grønn blå. Her går vi fra 0 til 255. rgb gir mer fleksibilitet siden vi kan bruke rgba som vi kan da assigne gjennomsiktighet. Vi kan også style i HTML tag typ h1 style = «color: pink» som da vil være at vi får en rosa overskrift. Derimot den mest hensiktsmessige løsningen er å danne forskjellige css filer. Dette er fordi hvis du eventuelt danner en nettside så vil du helst kunne designe flere like eller tilnærmet lik og da vil du kunne gjenbruke mye kode. Da vil det være mer syntax tungt å gå inn i hver html fil å endre eventuelt i header eller i hvert element. Da kan vi for eksempel skrive body har den layouten. Hvis vi ønsker å endre på den layouten for h1 så skriver vi h1 etter. Da vil h1 overskride siden den spesifiserer for et enkelt element. HTML elementer arver stil fra elementer de er nøstet inne i, typiske div eller p elementer. Tabeller arver ikke automatisk, du må bruke inherit for å spesifisere at den skal arve. Websikre skrifttyper er ofte det man bør bruke, og eventuelt ha flere som font type på nettsiden. Dette er fordi vi ønsker at vi skal ha muligheten til å representere teksten, og ved websikre så vil de være installert sammen med operativsystemet. Webfonter legges ved nettsiden omtrent som et bilde, der vi med filtre i CSS kan manipulere utseendet til bilder og annet innhold. JPG egner seg til fotografier, der GIF og PNG egner seg til logoer. #idnavn er ofte brukt som en hashtag selektor der vi velger et element med en bestemt id. .classnavn er en punkt selektor som velger alle av den klassen. Psuedoselektor velger derimot deler av et HTML element, og psuedoklasser velger elementer som er i en bestemt tilstand. Her er det ofte nyttig å bruke button:hover for knapper for å når man holder over knappen så kommer det en forstørring eller endring om ønsket som man spesifiserer i den klassen da.

Tagger:

body {}: Brukes til å style hele nettsidens kropp.

Ofte brukt: background-color, color, font-family, margin, padding.

div {}: Brukes til å style div-elementer (generelle containere).

Ofte brukt: margin, padding, border, background-color.

p {}: Brukes til å style avsnitt.

Ofte brukt: line-height, text-align, color.

h1, h2, h3, h4, h5, h6 {}: Brukes til å style overskrifter.

Ofte brukt: font-size, color, text-transform.

a {}: Brukes til å style lenker.

Ofte brukt: color, text-decoration, hover-effekter med :hover.

ul {} og ol {}: Brukes til å style lister.

Ofte brukt: list-style-type, padding, margin.

li {}: Brukes til å style listeelementer.

Ofte brukt: margin, padding.

table {}: Brukes til å style tabeller.

Ofte brukt: border, border-collapse, width.

tr {}: Brukes til å style rader i tabeller.

Ofte brukt: background-color, height.

td {}: Brukes til å style celler i tabeller.

Ofte brukt: padding, text-align, border.

th {}: Brukes til å style overskriftceller i tabeller.

Ofte brukt: font-weight, background-color.

form {}: Brukes til å style skjemaer.

Ofte brukt: margin, padding.

input {}: Brukes til å style inndatafelt.

Ofte brukt: border, padding, width, type.

button {}: Brukes til å style knapper.

Ofte brukt: background-color, color, border, hover med :hover.

label {}: Brukes til å style tekst for skjemaelementer.

Ofte brukt: color, font-size.

img {}: Brukes til å style bilder.

Ofte brukt: width, height, border-radius.

Vanlige CSS-egenskaper som brukes:

width: Setter bredden til et element.

height: Setter høyden til et element.

margin: Ytre avstand rundt et element.

padding: Indre avstand innenfor et element.

background-color: Setter bakgrunnsfarge.

border: Setter kantlinjer på elementer.

color: Setter tekstfarge.

font-size: Setter skriftstørrelse.

font-family: Angir skrifttype.

text-align: Justerer tekst (f.eks. venstre, høyre, midt).

display: Kontrollerer oppsett (f.eks. block, inline, flex).

position: Angir elementposisjon (f.eks. relative, absolute, fixed).

z-index: Angir lagrekkefølge (høyere verdi vises øverst).

overflow: Kontrollerer innhold som går utenfor boksen (f.eks. hidden, scroll).

box-shadow: Lager skygger rundt elementer.

border-radius: Gjør kantene på elementer runde.

Border-collapse: velger tabellens kantlinjer, seperate eller collapse (slå sammen)

Kap 3:  
Alle nettsider skal være universelt utformet slik at alle kan bruke nettsteder uavhengig av funksjonsevne. Kravene til universell utforming deles inn i 4 prinsipper; 1. Mulig å oppfatte. 2. mulig å betjene. 3.Forståelig og 4. Robust.

1 –

Mulig å oppfatte går ut på at informasjon og brukergrensesnitt må presenteres for brukere på måter som de kan oppfatte. Innholdstyper som bilder og lyd kan bare oppfattes av en sans og ikke alle, derfor for å kunne se bilder trenger vi en skjerm og synssansen og lyd høyttaler og hørselssansen. WCAG, Web content accessibility guidelines krever at vi har tekst alternativ til bilde. For da for å støtte folk med nedsatte funksjoner. Vi er også nødt til å sørge for å ha et kontrastforhold større enn 4,5: 1 siden ellers vil vi slite med å kunne tyde teksten.

2 –

Mulig å betjene går ut på at det må være mulig å forstå informasjon og betjening av brukergrensesnitt. Det er viktig at brukere kan navigere og velge knapper osv med det utstyret de har. Det skal være mulig at all funksjonalitet er tilgjengelig med tastatur.

3 –

Forståelig er at det må være mulig å forstå informasjonen og betjening av brukergrensesnitt. Målet med nettsider er vanligvis at brukerne skal forstå både hvordan de brukes og skjønne informasjonen de finner. Prinsippet handler om forutsigbarhet, enkelt språk og god hjelpefunksjonalitet. Her kan f.eks. sidespråk være oppgitt for å teksten blir lest opp på rett måte for dem som bruker talesyntese.

4 –

Robust er at innholdet må være robust nok til at det kan tolkes på en pålitelig måte av nettlesere og kompenserende teknologi. Dette prinsippet omhandler koding og at tilgjengelighet må ivaretas når man introduserer ny teknologi. I praksis betyr det at nettsider validerer at koden er riktig. Sette er som regel ivaretatt av standardelementer i HTML.

I Norge er det digitaliseringsdirektoratet som setter disse linjene.

Et bilde som inneholder skjermbilde, tekst, diagram, line

Automatisk generert beskrivelseWireframe er en enkel skisse som viser plassering av innholdet på skjermen. Når du skal planlegge en nettside kan en slik skisse være et godt utgangspunkt. I skissen tegner vi opp hvor navigasjonen, tekster, knapper og annet innhold på siden skal være. Bilde er ramme med kryss. Wireframe er hensikstmessig når vi skal designe nettsider siden vi kan fremme en layout for en potensiell kunde og diskutere før vi begynner kodingen.

Brukergrensesnittet til et nettsted sier noe om hvor brukervennlig nettstedet er, derfor når du skal desgine et nettsted er du nødt til å ta hensyn til at disse kravene:

* Enkle er bedre enn kompliserte
* Alle skal kunne forstå og bruke nettsiden
* En forside bør være oversiktlig
* Bruk bilder som illustrer innholdet
* Tenk på valg av tekst, ift størrelse, farge og linjelengde
* Nettsiden skal fungere på store og små skjermer

Filstruktur bør man ta hensyn til i form av at nettsidene dine bør ligge i en mappe, så er det kanskje greit at bilder ligger i en annen mappe for å ha en ryddig struktur. For inndeling på nettsiden bruker vi følgende:

Header – topptekst.

Main - hovedinnhold

Footer - bunntekst

Nav - navigasjon

Section – samler innhold som hører sammen

Article – selvstendig innhold

Div – diverse

Størrelser på nettsider bør vi ta hensyn til, dette skjer i form av at vi bruker noe som heter rem. Rem står for relativ em, og tilsvarer bredden av en stor M. 1 rem er standard skriftstørrelse som er 16 piksler. Det er vanlig å gjøre om skriftstørrelsen til 62.5 % siden da er 1 rem 10 piksler. Vw og vh (viewport width/height) er størrelsesenheter som tar utgangspunkt i nettleser vinduet. 50vw og 50vh tilsvarer halvparten av bredden og høyden til nettsiden. Vi bruker flex-box når vi skal plassere elementer i én dimensjon (ved siden av, eller under). En flex box inneholder flex items. I section selektoren skriver vi koden display: flex for å plassere flex items forskjellig. Med grid deler vi opp nettsiden eller deler av den i et rutenett, der vi kan endre på bredden ved fr. Et element som settes til 2 fr tar dobbelt så stor plass som et element i 1 fr.

Kap 4:

Nesten alt av digitale utstyr er datamaskiner som er bygd opp av de samme hovedkomponentene. I en datamaskin finner du blant annet hovedkort, harddisk (sekundærminne), prosessor (CPU), minnebrikke (RAM), grafikkort, nettverkskort, strømforsyning og USB-port.

Når vi skrur på datamaskinen starter et dataprogram BIOS, basic input output system. BIOS hjelper med å starte opp og få kontakt med hovedkort, harddisk, skjerm tastatur osv. BIOS starter også operativsystemet. Operativsystemet er f.eks. Apple OS og det tar over for BIOS når startet opp. Operativsystemet har som oppgave å kontrollere og styre kommunikasjonen mellom enhetene i datamaskinen og programmene som kjører. Operativsystemet sørger for at programmene får tilgang til enhetene og sikrer kommunikasjonen. Drivere sørger for at OS kan snakke til nettverkskort.

Hovedkortets oppgave er å sørge for kommunikasjon mellom de andre komponentene. Det er nødvendig for at maskinen skal fungere. Interne kontakter på hovedkortet kobler sammen prosessor minne lagring strøm og PCI Express. Prosessoren er festet rett på hovedkortet, har en vifte for å unngå å bli for varm. Minnebrikker og PCI er festet i egne spor på hovedkortet, mens strømforsyningen kobles til med en kabel.

Prosessor (Central processing unit) styrer og utfører instruksjoner fra programmene som kjører. De fleste instruksjonene utfører prosessoren selv, men den kan delegere andre spesialiserte oppgaver til f.eks. grafikkort. Prosessoren har et lite men svært raskt tilgjengelig innebygget cache. Der ligger det en kø med oppgaver når prosessoren er opptatt. Tidligere var minne USB og grafikk håndtert i egne fysiske brikker på hovedkortet, men nå er flyttet i prosessoren. I mobile enheter er det vanlig med SoC (system on a chip), da samles flere sentrale kontrollbrikker i prosessoren og færre styringsbrikker på hovedkortet. Da sparer vi mer plass og datamaskinen bruker mindre strøm til overføring av signaler mellom komponentene.

Harddisk er en fellesbetegnelse på de komponentene som i en datamaskin som lagrer filer over tid. De stasjonære datamaskinene og klumpede laptopene hadde magnetdisker (HDD – hard disk drive) som da var overlegne på grunn av lagringskapasitet, overføringshastighet og stabilitet. Etter hvert modnet teknologien som brukes i minnekort. Da ble SSD (solid state drive) videreutviklet og vanlige datamaskiner kan utnytte fordelene med høyere skrive og lese hastigheter. SSD gjør det mulig å lagre store mengder informasjon på små minnebrikker med et veldig lavt strømforbruk, og teknologien brukes i mange laptoper og alle mobile enheter.

RAM

Kode og filer som kjøres blir liggende i arbeidsminnet mens de er i bruk. RAM (random access memory) er normalt ikke større enn noen GB, men det arbeider mange ganger raskere enn en harddisk og er stort nok til å lagre de filene vi arbeider med. Det er delt opp i en rekke plasser hvor lesing og skriving kan gjøres uten å påvirke dataene i de omliggende plassene. Dataene deles opp over flere plasser i minnet uavhengig av hverandre.

Grafikkort

I alle datamaskiner som viser info på en skjerm fins det et grafikkort som effektivt tolker og behandler instrukser fra maskinen og omformer dem til et bilde som kan vises på skjermen. Prosessoren i maskinen er utstyrt for å kunne behandle et bredt spekter av instrukser og koder etter hverandre. Styrken er at den kan behandle ting samtidig, dette gjøres fordi grafikkprosessoren kan behandle flere instruksjoner parallelt.

Nettverkskort

For å kunne kommunisere med omverdenen er det nødvendig med et nettverkskort. I dag har vi trådløse nettverkskort, før utvidelsesspor på hovedkortet, brukte nettverkskabel fra uttak i veggen. Nettverkskort omhandler ethernet, wifi, mobile nettverk 4g 5g og blåtann.

PCI Express er en høyhastighets tilkoblingsstandard som brukes til å koble forskjellige komponenter i en datamaskin, som grafikkort, lagringsenheter osv. til hovedkortet.

Vi bruker binære tall 1 og 0 for å bestemme om tilstandene i en datamaskin er av eller på. Så hvis vi tar da logikken i det så vil forskjellige logiske porter gi oss forskjellige utsagn og bestemme forskjellig når av og på gjelder. To bit gir 4 muligheter, tre bit gir 8 osv. Dette er fordi bit representeres med 1 og 0, og det binære tallsystemet går etter totalsystemet, der vanlige tall går etter 10 tallsystemet. 1 byte = 8 bit, så for 28 = 256, som gir 256 muligheter. Dette er stort nok for å representere en farge, et lydnivå eller til å angi et tegn i et tegnsett. Dette er fordi vi ikke kan representere tegn med bokstaver, så for å gjøre det bruker man f.eks ASCII. Der får man 128 muligheter og er 7 bit, men i dag bruker vi Unicode som gir oss 256.

Et bilde deles inn i bitte små deler som kalles piksler eller bildepunkter. Disse bildepunktene har ikke noen fast eller standardisert størrelse. Desto flere punkter du deler et bilde inn i, desto høyere blir oppløsningen på det. Farger kan vi som tidligere beskrive av RGB eller CMYK verdier.

**RGB**: Står for **Red, Green, Blue** og brukes til å definere farger ved å angi intensiteten av rød, grønn og blå (fra 0 til 255).

**RGBA**: Samme som RGB, men inkluderer en **Alpha-kanal** for gjennomsiktighet (fra 0 til 1). Eksempel:

**Regler:**

1. Heksadesimale tall bruker basen 16.
   * A = 10, B = 11, ..., F = 15.
2. Hver heksadesimal består av to deler:
   * Den første posisjonen har en vekt av 16^1.
   * Den andre posisjonen har en vekt av 16^0.

**Eksempel 1:**fc**til desimal**

1. fc betyr:
   * f = 15, c = 12 i desimal.
2. Kalkulasjon:

(15×161)+(12×160)

=(15×16)+(12×1)

=240+12

=252

**Eksempel 2:**a5**til desimal**

1. a5 betyr:
   * a = 10, 5 = 5 i desimal.
2. Kalkulasjon:

(10×16)+(5×16)

=(10×16)+(5×1)

=160+5

=165

**Eksempel 3:**02**til desimal**

1. 02 betyr:
   * 0 = 0, 2 = 2 i desimal.
2. Kalkulasjon:(0×161)+(2×160)

=(0×16)+(2×1)

=0+2

=2

1. **Oppsummert:**

* fc = **252**
* a5 = **165**
* 02 = **2**

Dette gir rgb(252, 165, 2)

Mediefiler (bilde, lyd og video) tar stor plass. For at datafilene skal ta mindre plass uten at de blir ødelagt og uten at vesentlig informasjon i for eksempel et bilde går tapt bruker vi kompresjon. Tapsfri kompresjon er når vi har gjort en fil mindre i størrelse, men likevel har nok data til å gjenskape en fil som er identisk med originalen. Destruktiv kompresjon brukes ofte når det gjelder mediefiler f.eks. bilder og er at vi fjerner detaljer som vi med våre øyne sliter med å klare å se og kan representere et godt nok bilde til en mye lavere verdi. Tegnsettet i en datamaskin avgjør hvordan bokstavene i alfabetet, tall og andre skrifttegn skal vises på skjerm. Lyd er hurtige endringer i trykk, kalt lydbølger. Skal vi lagre og behandle lyd må lydsignalet digitaliseres, altså tall. Video kan komprimeres ved bruk av to metoder: Run length encoding (hvert bilde behandles for seg, datamengden reduseres uten kvalitetstap) og JPEG komprimering (hvert bilde komprimeres med en JPG algoritme).

Et nettverkskort gjør at datamaskiner kan kobles opp mot et nettverk av andre enheter.

URL – Uniform resource locator

HTTP – Hyptertext transfer protocol

DNS – domain name system (hierarchical, root (peker til tld), TLD domain(.com, .no osv, peker til authoriative), authoritative (ip adressen til example.com))

Kap 5:

Javascript er et programmeringsspråk som blir oversatt til maskinkode av netteleseren. Vi bruker Javascript når vi lager dynamiske nettsider. Javascript kode skrives inne i et script element.

I en dynamisk nettside kan du utføre en handling som påvirker utseendet eller innholdet på nettsiden. Slike handlinger kan være klikke, tappe, sveipe taste inn laste opp bilder etc. En kompilator oversetter høynivå språk til maskinkode og motsatt.

En variabel består av et navn og en verdi og defineres slik i script elementet:

let navn = verdi.

Hvis verdien til en variabel er en tallverdi, skriver vi bare tallet. I desimaltall bruker vi punktum som desimaltegn. Variabelen blir da av datatypen number.

Hvis verdien til en variabel er tekst, skriver vi det med apostrofer rundt teksten «tekst-verdi». Variabelen blir da av en type kalt string, å skjøte sammen to eller flere tekster gjøres med + tegnet. Dette kalles konkatenering. Så «fri» + «minutt» = «friminutt».

Du kan sjekke datatypen til en variabel med typeof etterfulgt av variabelnavnet.

Når vi vil hente frem verdien til variabel på nettsiden vår skriver vi krøllparentes rundt variabelens navn slik:

<p> min gode venn, {navn}! </p>.

Når vi skriver <input bind: value = {navnpåVariabel}> oppdateres verdien til variabelen med det som blir skrevet i innskrivningsfeltet. Dette er databinding.

I Javascript finnes det innebygde metoder for datatypen string og matematiske metoder:

String:

Matematiske metoder:

Reaktive variabler oppdateres når én eller flere variabler de er utledet fra, endrer seg. Vi definerer den reaktive variabelen areal slik:

$: areal = lengde \* bredde.

Boolske verdier er verdier som enten er true eller false. Boolske uttrykk sammenligner verdier og gir svar på om uttrykket er sant eller usant.

I script elementet skriver vi en if setning slik:

If (test) {

//gjør noe

}

else if (en annen tekst){

//gjør noe annet

}

Else {

//gjør noe helt annet

}

I HTML skrives det slik:

{# if test1}

<p>Innhold vises hvis test1 er true</p>

{: else if test2}

<p>Innhold vises hvis test2 er true og test 1 false</p>

{:else}

<p>Innhold vises hvis test1 er false og test2 er false</p>

{/if}

En funksjon som består av én eller flere linjer med kode. Funksjoner kan ta input-data som kalles parametere og returnere én verdi. Koden for å lage en funksjon er på denne formen:

const funksjonsnavn = {parameter1, parameter2} => {

//Skrive en eller flere linjer med kode

Return variabelNavn

}

Koden kjøres når vi kaller på funksjonen med funksjonsnavn(). Hvis vi vil at funksjonen skal kjøres når en knapp trykkes på skriv;

<button on:click = {funksjonsnavn}>

Vi har noe som heter en anonym funksjon, en anonym funksjon er en funksjon uten noe navn. En vanlig situasjon hvor vi får bruk for en anonym funksjon er hvis vi ønsker å kalle på en funksjon med parametere med klikk på en knapp eller lignende. Koden for anonym funksjon er:

() => {kode som skal utføres.}

Kap 6:

Et array er en variabel med flere verdier. Vi definerer et array slik:

let arraynavn = [verdi1, verdi2, verdi3].

Verdiene i et array nummereres med indekser der den første verdien har indeks 0, den neste 1 osv. Helt til det ikke fins flere verdier.

For å hente ut verdien i et array som ha indeks 2 skriver vi arraynavn[2].

Et objekt består av ett eller flere nøkkel par. Vi oppretter et objekt slik:

let objektnavn = {

nøkkel1 : «Verdi 1»,

nøkkel2 : «Verdi 2»

}

Hvert nøkkelpar separeres med et komma og alle parene er omsluttet krøllparenteser.

For å hente ut en verdi fra et objekt bruker vi dot-notasjon slik, objekt.nøkkel1.

Vi kan lagre data ved å bruke et array som inneholder objekter:

let objektnavn =[

{nøkkel1 : «Verdi 1»,nøkkel2 : «Verdi 2»},

{nøkkel3 : «Verdi 3»,nøkkel4 : «Verdi 4»}

]

En each-blokk i Svelte går gjennom et array element for element. Vi skriver en each-blokk slik:

{#each arraynavn as navn}

{navn.nøkkel1} {navn.nøkkel2}

{/each}

JSON star for JavaScript Object Notation. JSON -filer inneholder objekter. Det er et filformat som de fleste programmeringsspråkene kan lese.

Når vi skriver JSON må vi bruke apostrofer rundt nøklene. Når vi definerer et objekt i js skjer dette som regel at vi definerer variabel eller konstant som symboliserer verdien til objektet. I JSON starter og slutter et objekt med krøllparenteser og et array med klammeparenteser. For å hente data fra en JSON fil bruker vi i js noe som heter fetch. Da henter vi ut data fra JSON filer og andre tekstfiler. JSON hentes som regel fra andre steder og vil derfor ta lang tid å hente.

JSON filer kan kun ha en datastruktur. Det betyr at hvis du skriver en JSON fil kan du kun enten ha array typer eller kun objekt typer.

En asynkron funksjon kan vente med å gå videre til en hendelse har inntruffet, for eksempel at data fra en fil er lastet inn i nettleseren. Det er bare i en asynkron funksjon at det er lov å bruke await. Oppskriften for fetch er:

1. Skriver inn kodeordet async før parameter parentesene til funksjonene
2. Vi «henter» inn filen som inneholder data med fetch («./filnavn»)
3. For at kjøringen skal vente på at vi får lastet inn data skriver vi await før fetch («./filnavn»).
4. fetch kan hente andre typer data enn JSON. Derfor må vi i koden vår fortelle nettleseren at dataene vi skal ha tak i er formatert som json. Da skriver vi respons.json()

Et data-API (Application Programming Interface) er en tjeneste som tilbyr data som vi kan bruke på for eksempel nettsider. Grensesnittet (interface) definerer hvordan et dataprogram kan motta eller sende data til et annet dataprogram, nettside eller app.

Med localStorage kan nettleseren lagre data som er unike for én type nettleser på én spesifikk enhet. Med localStorage lagres dataene persistent. Da er de varige og forsvinner ikke når du lukker nettleseren.

Med sessionStorage lagres dataene så lenge nettleseren er åpen. Hvis du lukker nettleseren eller den fanen som nettsiden er åpen i, vil dataene slettes.

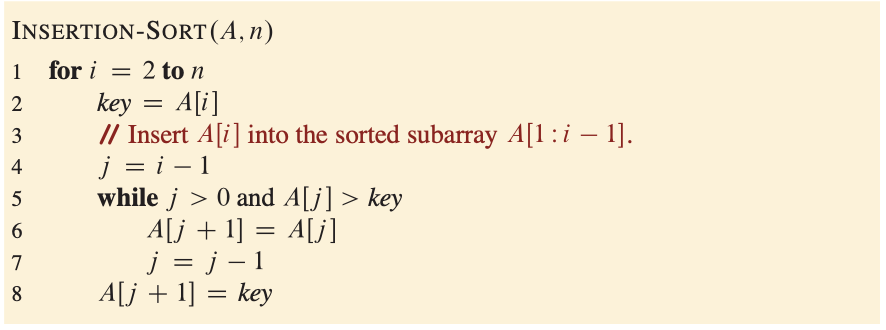
Praktisk bruk av lokal lagring:

* Lagring av data ved registrering av opplysninger, da blir det slik at det du har skrevet inn ikke forsvinner hvis du skulle komme borti tilbakeknappen eller lignende.
* Å sette utseendet til nettsiden. Du kan selv velge hvordan nettsiden skal se ut. Neste gang du åpner den vil den fortsatt ha samme utseende.
* En nettside for feriebestilling som husker hvilke datoer og reisemål du søkte på forrige gang.

I databaser lagres store mengder data på en strukturert måte, databaser er f.eks Firebase fra Google og Supabase som er basert på åpen kildekode. Åpen kildekode betyr at programvarekoden er fritt tilgjengelig for alle, slik at den kan leses, endres, og distribueres av hvem som helst.

Kap 7:

En algoritme er en stegvis oppskrift for å løse et problem. Algoritmen har ofte inndata og utdata. Vi bruker algoritmisk tenkning når vi løser problemer. Når vi bryter ned et stort problem til to eller flere mindre biter, dekomponerer vi problemet.



Abstraksjon er å finne en måte å se en gruppe objekter eller ideer på, blant alle mulige måter å se disse, som gjør at vi kan løse et problem.

Å finne mønstre er å se sammenhenger innad i en oppgave og mellom oppgaver som hjelper oss med å løse noe vi holder på med. Innenfor algoritmisk tenking er logikk kunsten å uttrykke mønstre man har sett ved hjelp av logiske uttrykk som inneholder operatorer som <, > ===, !==, && og | |. Etter at vi har laget og implementert en algoritme som løser et problem må vi evaluere den. Her sjekker vi at algoritmen faktisk løser problemet den skulle løse og om koden er korrekt. Når vi har søkt etter fil og test algoritmen må relevante inngående data, bør vi også se etter muligheter for å effektivisere koden, slik at den ikke bruker unødvendig lang tid på å kjøre.

Tips for feilsøking er å bruke console.log, console.table og bruk beskrivende navn på variabler og funksjoner. Du bør også unngå mutasjoner og sikre deg mot endringer på et array med spredningsoperatoren. Hvor mange steg en algoritme går gjennom, er et mål for hastigheten til algoritmen. Vi må være oppmerksomme på antall steg i løkker i koden vi skriver selv, men også når vi bruker innebygde metoder som inneholder løkker. For å øke hastigheten kan vi prøve å flytte noen steg ut av løkker ved å lage variabler eller funksjoner.

Utforsking er prosessen med å bli kjent med et problem og hvordan vi skal løse det. Ferske programmerere trenger å utforske programmeringsspråket i tillegg til problemet i seg selv. Vi kan bruke programmeringsspråkets dokumentasjon i utforskningen.

For at noe skal kunne kalles en algoritme må den oppfylle følgende krav:

1. En algoritme er en prosess som du mater noe inn i og får et resultat tilbake fra.
2. Prosessen må være beskrevet stegvis. Hvert steg må være så entydig beskrevet at personen uten tvil vil kunne gjennomføre det og ende opp med samme resultat
3. Hvert steg må tydelig lede videre til neste
4. Med samme inndata skal algoritmen få samme utdata
5. Algoritmen skal helst stoppe en eller annen gang.

Kap 8 og Kap 9

I løpet av et tiår har sosiale medier forandret måten vi kommuniserer med hverandre på. Vi går i retning av et mer teknologibasert samfunn og Norge blir ofte omtalt som ett av de mest digitale samfunnene i verden. Ungdom bruker opp mot 7 timer i gjennomsnitt hver dag på nettet. Ungdom bruker mindre tid sammen med venner ansikt til ansikt og mer tid på Internett og sosiale medier. Varig lagring: kommunikasjon i sosiale medier blir lagret på nett i all fremtid. Digitale identiteter er søkbare. Gjenbruk, vi kan på nettet klippe og lime et budskap slik at det kan være vanskelig å skille kopien fra originalen. Usynlige tilskuere på nettet har vi i mindre grad kontroll på hvem som ser hva og hva vi gjør. Men også med sosiale medier får vi hatt lettere kontakt med andre med samme språk etc.

Nettregler:

* Vær mot andre på nettet slik du vil de skal være mot deg
* Beskytt navnet ditt, lag kallenavn
* Gi ALDRI ut passord
* Forsiktig med opplysninger om deg selv
* Spør om lov ved opplysninger av andre
* Sjekk alltid hvem som har laget informasjonen du finner
* Tenk deg om før du laster ned innhold.

Vi trenger digitale tjenester for å få samfunnet til å fungere. Dette har gjort oss mer sårbare for nye typer trusler. Det som er spesielt med den digitale sårbarheten er at når feilene først utløses rammer de flere områder samtidig. I det digitale rom er vi utsatt for mange forskjellige trusler aktører; hacktivister, kriminelle etc. De tre hovedpunktene i informasjonssikkerhet er konfidensialitet, integritet og tilgjengelighet. Alle som arbeider med informasjonsverdier, har ansvar for å beskytte dem. Den norske staten har et overordnet ansvar å beskytte de funksjonene og systemene som landet er avhengig av, dette er det vi kaller kritisk infrastruktur. Nasjonal sikkerhetsmyndighet hevder at hvis alle virksomheter hadde fulgt deres fire grunnleggende råd:

1. Oppdater program- og maskinvare

2. Installer sikkerhetsoppdateringer så fort som mulig

3. Ikke gi sluttbrukere admin-rettigheter

4. Blokker kjøring av ikke autoriserte programmer

Ville mange hendelser vært unngått. Personvern handler om at våre personlige data skal beskyttes og et digitalt grenseforsvar vil bestå av en «føre-var lagring». Datalagringsdirektivet sier at datatrafikk og informasjon fra tlf, epost, mobil og internett skal lagres slik at opplysningene kan bekjempe terror og kriminalitet. Problemet er at dette bryter menneskerettighetene om ytringsfrihet. Edward Snowden publiserte hemmelig informasjon om etterretningsprogrammet til USAs etterretningsinnsamling fra Internett. Snowden mente at etterretninga hadde blitt for omfattende. Personvern handler om retten vår til å velge, vite og ha kontroll med hva som skjer med dataene våre. Personopplysningsloven er det som regulerer behandling av data som ikke er regulert av et annet lovverk. Vi har tre kategorier av personopplysninger: (1. Data som ikke identifiserer personer, 2. Data som identifiserer personer, 3. data som identifiserer personer og er sensitive. )

En bruker skal få vite hvem som behandler dataene, hvilke data som samles inn, hva de brukes til og hvor lenge de lagres og brukes.

EKOM loven er loven som regulerer elektronisk kommunikasjon, tlf bruk sms, bruk av bredbånd osv.