### Oppgave 1: Kalkulator, Forklaring av feilsøking, forståelse av funksjoner og utskrift

Planlegging:

Funksjon for å vise tallene, funksjon for beregningen, funksjon for å slette alt fra forrige beregningen , funksjon for å huske utberegninger , funksjon for å gå gjennom alle tallene frem og tilbake som ble lagt for å huske og til slutt en funksjon som aktiverer / deaktiverer knapper for bla gjennomminne basert på minen

funksjoner:

display(val):

Denne håndterer visning av tall og operatører i inntastingsfeltet.

Logger den viste verdien for feilsøking.

**funksjonen "solve()"**

Bruker eval for å evaluere den matematiske uttrykket og vise resultatet."solve" funksjonen er designet for å løse et problem eller utføre en beregning. For eksempel, hvis du ser denne koden i sammenheng med matematikk eller lignende, kan "solve()" være ansvarlig for å løse en ligning eller utføre en beregning for å finne en løsning.

Variabler som memoryBank og currentMemoryIndex brukes til å lagre tilstand og dele informasjon mellom funksjoner.

clearScreen():

Tømmer inntastingsfeltet.

addMemory():

Legger til gjeldende beregning i minnebanken.

Oppdaterer antall elementer i minnebanken og gjeldende minneindeks.

Aktiverer minneknapper.

browseMemory(retning):

Tillater navigasjon gjennom minne i både fremover og bakover retning.

Oppdaterer vist beregning fra valgt minne.

Deaktiverer/Aktiverer minneknapper etter behov.

updateMemoryCount():

Oppdaterer visningen av antall elementer i minnebanken.

updateCalculationFromMemory():

Oppdaterer inntastingsfeltet med valgt minneelement.

Aktiverer/Deaktiverer minneknapper etter behov.

enableMemoryButtons():

Aktiverer eller deaktiverer minnenavigasjonsknapper basert på gjeldende minneindeks.

Hver funksjon har et klart definert ansvar. For eksempel, display, solve, clearScreen, addMemory, etc., har spesifikke oppgaver i kalkulatoren.

Konsollogging:

Brukte console.log for å logge informasjon under utvikling er en god praksis for feilsøking.

Kommentarer:

Noen kommentarer er inkludert for å forklare hensikten med visse deler av koden i html. Dette er for sensoren eller andre som skal lese koden

Kilder som ble brukt som hjelpe midler til å løse

<https://copyprogramming.com/howto/javascript-clear-screen-function-in-javascript#clear-calculator-display-javascript-after-calculation>

<https://www.geeksforgeeks.org/clear-console-c-language/>

<https://www.w3schools.com/jquery/html_val.asp>

<https://docs.appian.com/suite/help/23.4/fnc_conversion_displayvalue.html>

<https://developer.mozilla.org/en-US/docs/Web/API/setTimeout>

<https://www.geeksforgeeks.org/jquery-val-method/>

<https://stat.ethz.ch/R-manual/R-patched/RHOME/library/base/html/solve.html>

<https://doc.sagemath.org/html/en/tutorial/tour_algebra.html>

<https://www.makeuseof.com/build-a-simple-calculator-using-html-css-javascript/>

<https://www.educative.io/answers/how-to-build-a-calculator-in-javascript>

<https://stackoverflow.com/questions/73746625/button-onclick-function-adds-class-to-div>

<https://www.geeksforgeeks.org/javascript-calculator/>

<https://stackoverflow.com/questions/48045613/javascript-calculator-equal-function-failing>

Oppgave 2: Randomisert tegning på canvas

Planlegging for Oppgave 2

Settes opp canvas-dimensjoner og kontekst.

Legges til en firkant eller sirkel ved og håndtering klikk på S-knappen eller C-knappen og tilfeldig plassering av elementene

Fjernes et tilfeldig element fra canvasen ved å trykke på R-knappen.

Oppdatering av gjennomsnittlig antall elementer per sekund.

Oppdatering brukeraktivitetstiden.

Oppdatering antall firkanter, sirkler og totalt antall elementer på grensesnittet.

Tegner elementene på canvaset ved hjelp av requestAnimationFrame.

Tegner firkantene og sirklene på canvasen basert på type elementer.

Legges til event listenes for tastetrykk (S, C, R) og klikk på tilsvarende knapper.

Brukes knapper og tastetrykk for å legge til og fjerne elementer.

Oppdater tellingen av elementer på grensesnittet.

Brukes requestAnimationFrame for å oppdatere tegningene og sikre jevn animasjon.

Canvas-elementet er hentet ved hjelp av document.getElementById("drawArea").

Bredde og høyde på Canvas er satt til 600x500 piksler.

Canvas-konteksten (ctx) blir brukt til å tegne på Canvas.

Elementer:

elements-arrayet inneholder objekter som representerer firkanter og sirkler på Canvas.

Objektene har egenskaper som type (square eller circle) og koordinater (x og y).

Funksjoner:

addElement(type): Legger til et nytt element (firkant eller sirkel) til elements-arrayet med tilfeldige koordinater.

removeRandomElement(): Fjerner et tilfeldig element fra elements-arrayet.

updateCounts(): Oppdaterer grensesnittet med antall firkanter, sirkler og totalt antall elementer.

Event Listeners:

Lytter etter klikk på knapper (S, C, R) og tastetrykk (s, c, r) for å utføre tilsvarende handlinger.

Rendering:

renderDrawings(): Rydder Canvas og kaller drawElements() for å tegne elementene.

drawElements(): Går gjennom hvert element og tegner enten en firkant eller sirkel basert på typen.

Oppdatering og animasjon:

updateCounts() kalles for å oppdatere antall elementer i grensesnittet.

requestAnimationFrame(renderDrawings) brukes for å kontinuerlig oppdatere tegningen i en animasjonssløyfe.

Start Drawing:

Siste del av koden starter tegneprosessen ved å kalle requestAnimationFrame(renderDrawings).

Kommentarer:

Noen kommentarer er inkludert for å forklare hensikten med visse deler av koden i html. Dette er for sensoren eller andre som skal lese koden

# Oppgave 3: Ansatregister (Vekt: 50%)

Planlegging:

Mål: Registrere ansatte med informasjon som fornavn, etternavn, stilling, avdeling og e-postadresse.

Vise ansatte i en liste , Lages JavaScript-funksjoner for å fylle inn staff-listen med data fra staff-arrayet.

Legges til nye stillinger og avdelinger.

Lages funksjoner for å fylle inn dropdown-menyene for position og department med data fra positions og departments-arrayene

lages en funksjon for å legge til nytt personale basert på inndata fra skjemaet og oppdateres ansatt listen

Legges til ny stillinger Implementer en funksjon for å legge til en ny stilling basert på inndata fra skjemaet oppdateres avdelinger etter det

Legges til event listeners for knapper som "Save staff member," "Save new position," og "Save new department" for å utløse tilsvarende

JavaScript-funksjoner.

renderStaffList(filteredPosition): Oppdaterer visningen av ansatte basert på filtrering.

renderPositions(): Gjengir stillinger i posisjonsmenyen.

renderDepartments(): Gjengir avdelinger i avdelingsmenyen.

renderPosisionFilter(): Oppdaterer posisjonsfiltreringen basert på eksisterende stillinger.

Event listeners for lagring av ansatte, stillinger og avdelinger.

Her er det forklaring av totalt hvordan den registreres

Legg til ansatte:

Skriv inn fornavn, etternavn, velg stilling og avdeling, skriv inn e-post, og trykk på "Lagre ansatt."

Informasjonen legges til i ansattearrayet, og visningen oppdateres.

Legg til stilling:

Skriv inn stillingsnavn og trykk på "Lagre ny posisjon."

Stillingen legges til i stillingsarrayet, og menyen og filtreringen oppdateres.

Legg til avdeling:

Skriv inn avdelingsnavn og trykk på "Lagre ny avdeling."

Avdelingen legges til i avdelingsarrayet, og menyen oppdateres.

Filtrering:

Klikk på posisjonsfiltreringsknapper for å vise ansatte basert på valgt stilling.

"All positions" viser alle ansatte uavhengig av stilling.

Interaksjon med arrayer:

staff-array:

Inneholder objekter med ansattinformasjon.

Legges til når lagrer en ny ansatt.

positions-array:

Inneholder stillingsnavn.

Oppdateres når legges til en ny stilling.

departments-array:

Inneholder avdelingsnavn.

Oppdateres når legges til en ny avdeling.

Oppsummering:

Dette nettstedet gir en brukervennlig måte å administrere ansatte, stillinger og avdelinger på.

Kommentarer:

Noen kommentarer er inkludert for å forklare hensikten med visse deler av koden i html. Dette er for sensoren eller andre som skal lese koden