Arbeidskrav 2: Sammenligne kode

Oppgave 1:

Problem 1:

For å løse dette problemet krevde det generering av tilfeldig heksafargekode, da var min strategi å bekrefte den nødvendige JavaScript syntaksen. Jeg gjorde et google søk og fant en løsning som benyttet for-løkke kombinert med (Math.random) for tilfeldig indeksering.
Jeg brukte denne logikken ved å

1. Definere alle gyldige tegn i en streng.
2. Bruke en løkke som gjentar seks gnager for å bygge opp koden.
3. Konvertere det endelige resultatet til det riktige HTML-elementet.

Problem 2:

Jeg skjønte raskt at jeg måtte sjekke om bestemte ord eksisterte i en array. Med god kjennskap til array-metoder, visste jeg at jeg måtte bruke *if/else*- strukturen. Jeg brukte verktøy som **W3schools** for å bekrefte den nøyaktige syntaksen for if/else.

Problem 3:

Min løsning på dette problemet var å tenke på logisk matte. Når du må finne gjennomsnittet må du summere også dele på antall tall. Ved å bla opp i pensum i **Webtricks** innfor løkker fant jeg en oppskrift for å regne gjennomsnittet trinnvis inne i selve løkken.

1. Jeg lagde en variabel (let sum= 0) utenfor løkken
2. I *forEach-løkken* tok jeg hvert enkelt tall og delte det på arrayens lengde
3. Deretter la jeg det resultat til sum-variabelen

Oppgave 3:

Jeg brukte **Google Gemini** og **Chatgpt** for å løse denne oppgaven. Jeg gikk gjennom flere utfordringer. Jeg måtte bytte fra Google Gemini for den inkluderte CSS i javascripten koden noe som skapte feil. Jeg byttet over til Chatgpt og der var den bedre når det kom til kodene. Det ble litt surr i enden for den dupliserte kodene. Jeg måtte flere ganger be den rette.
Dette beviste at selv om KI kan gi raske løsninger, krever det menneskelig tilsyn og feilsøking.

Sammenligning av min kode og KI sin kode.

Problem 1	**Min kode**	**KI kode**
Forskjell	• Koden min kjøres umiddelbart linje for linje. Den gjør koden mindre gjenbrukbar. • Jeg bruker engelske variabel navn • Bruker. innerHTML	• KI koden har laget en gjenbrukbar løsning ved å bruke *function genererTilfeldigFarge()* • KI bruker norske variabelnavn • Bruker. textContent

Problem 2	**Min kode**	**KI kode**
Forskjell	• Koden min repeterte den samme logikken 3 ganger • Jeg bruker utvidede if/else strukturen	• Koden definerer logikken en gang og kaller den tre ganger • KI bruker (? :) for å komprimere logikken til en linje.

Problem 3	**Min kode**	**KI kode**
Forskjell	• Bruker forEach-løkke for å legge sammen summen,	• Bruker reduce () for å finne den totale summen først, og

	men deler hvert tall med lengden i løkken • Deler hvert tall og summerer deretter. Det gjør det mindre presis. • Krever flere linjer for å gjenta samme logikk for Array 1 og Array 2	deler en gang på lengden etterpå. • Den finner nøyaktig sum først, deretter deler. • Bruker kun to linjer kode i selve funksjonen.

Gir kunstig intelligens en mer forståelig løsning? Bruker KI ting du kjenner til, eller lærer du noe nytt ved å bruke den?

Jeg vil ikke si at KI sin kode ga en mer forståelig løsning. Den gir en mer profesjonell løsning, men den er ikke så enkelt å lese for en nybegynner vil jeg si. Men den brukte logiske navngitte funksjoner som *(genererTilfeldigFarge, sjekkOrd, finnGjennomsnitt)* noe som gjør det klart hva hver del av koden gjør, uten å måtte lese igjennom selve logikken hver gang. Det er veldig tydelig.

Den brukte mange ting jeg kjente til fra før av som *variabler, for-løkker og getElementById.* KI koden bekreftet min grunnleggende forståelse ved at kjernelogikken var lik, men den lærte med bedre JavaScript praksis ved å introdusere verktøy som reduce() og den avgjørende viktigheten av gjenbrukbare funksjoner og feilhåndtering.

I et tekstdokument, fortell hvilken/hvilke KI-verktøy du har benyttet. Beskriv hvordan du har tenkt og kodet annerledes enn KI, og hva du har lært av å sammenligne koden din og KI sin kode.

Jeg brukte KI-verktøyet **Google Gemini** for å løse oppgavene.

Når det gjelder **problem 1** (fargekode). Her tenkte jeg enkelt og rett frem. Lag tegnsett, start med # også gjenta seks ganger, finne tilfeldig tegn, skriv ut.

Når det gjelder **problem 2** (ordskjekking). Her tenkte jeg at jeg må sjekke hver for en. Dette resulterte i tre separate if/else-blokker som gjentok den samme logikken.

Når det gjelder **problem 3** (gjennomsnitt). Her visste jeg at jeg måtte summere og dele. Jeg valgte å bruke *forEach* og delte hvert enkelt tall med arrayens lengde.

Konklusjon: Min kode er forståelig for en nybegynner, men den gjentar seg selv og kan ses på som vanskelig å endre eller utvide.

Referanseliste:

GeeksforGeeks. (2025, 11. juli). *JavaScript program to generate random hex codes of color.* Hentet fra https://www.geeksforgeeks.org/javascript/javascript-generate-random-hex-codes-color/

Google. (2025). *Svar på spørsmål om hjelp til å besvare KI-oppgaven (koder med CSS)* [Gemini large language model output]. Hentet fra https://g.co/gemini/share/172eeaa84763

OpenAI. (2025). *KI-generert kode til bruk i oppgave 2* [ChatGPT large language model output]. Hentet fra https://chatgpt.com/share/68ee6577-827c-8008-b8d7-266616edf099

W3Schools. (u.å.). *JavaScript array includes() method.* Hentet fra https://www.w3schools.com/jsref/jsref_includes_array.asp

Webtricks. (u.å.). *Foreach-løkker.* Hentet fra https://lms.webtricks.blog/kurs/innforing-i-programmering/lokker/foreach-lokker