**Kjennetegn på god JavaScript kildekode**

1. Filene organiseres i hensiktsmessige mapper slik at programmet er lett å finne frem i også for andre enn den som har lag løsningen. I prøvesituasjoner lager vi som regel ei mappe per oppgave.
2. Programmet bygges opp etter en standard struktur: HTML (index.html), JavaScript(script.js) og CSS (stil.css) lagres i hver sine filer. Funksjonen winInit() har rolle som programmets hovedprogram og ligger i starten av programmet (etter programsetningen window.onload = winInit;. Globale variabler som benyttes deklareres først i script-filen.
3. Navnene på funksjoner og variable lages med omhu, slik at det er mest mulig selvforklarende hvilke rolle de har i programsystemet. Alle variables skal deklareres, lokale variabler i starten av funksjonen. Fra navnet på variablene er det en fordel om man kan forstå hvilke datatype de har. Vi benytter kamelskrift ved sammensatte ord.
4. Koblingene mellom JavaScript og nettsidens HTML minimaliseres mest mulig. Det betyr i praksis at det er hensiktsmessig å lage globale variabler som inneholder «håndtakene» til HTML-elementene (ved bruk av document.getElementById(«navn») – i winInit() ). En variabel som er et «håndtak» til et HTML-element følger konvensjonen <navn>\_HtEl (*eksempel bildeLaat\_HtEl* ).
5. Funksjoner benyttes aktivt for å sikre lett lesbar kode, færre kodelinjer, lett test av delsystemer og gjenbruk av koden i flere delsystemer.
6. Funksjoner generaliseres hvis mulig, slik at de lett kan legges inn i et verktøybibliotek og på den måten kunne anvendes i flere programsystemer. Slike verktøybibliotek legges i egen mappe. Generalisering av funksjoner innebærer at man unngår (om mulig) bruk av globale variabler og koblinger mot nettsidens HTML.
7. Når man henter verdier fra HTML må man i JavaScript alltid bruke casting dersom verdiene skal representeres som et tall [parseInt(tekst) leverer (om mulig) et heltall, parseFloat(tekst) leverer (om mulig) et desimaltall – Number(tekst) kan også benyttes]. Det er også god praksis å teste brukerinput og gi brukeren en tilbakemelding ved feilsituasjoner.
8. Kildekoden kommenteres kort og presist der det er nødvendig, slik at en annen utvikler lett kan settes seg inn i hva kildekoden gjør. Velg ett språk for kommentarene i koden (ikke miks norsk og engelsk).
9. Generelle funksjoner (i verktøykassebiblioteket) dokumenteres spesielt grundig. Må inneholde beskrivelse av; formål, inputargumenter, bruk av globale variable (inn, ut), evt. retur.
10. Koden skal bruke innrykk. Innrykkene skal vise til hvilke kodeblokk programsetningene tilhører og gjøre det lett å telle parenteser. Bruken av innrykk skal være konsekvent i alle deler av programsystemet.
11. Det utviklede programsystemet skal i ettertid være lettest mulig å vedlikeholde(feilretting) og forbedre. Det betyr at man unngår «hardkoding» der det er unødvendig (eksempel: bruk alltid lengde-attributten til en array i stedet for en fast verdi). Det betyr også at man skal vurdere å lage spesifikke funksjoner hvor innholdet lett kan generaliseres (eksempel: hentSpilleliste(); i stedet for faste verdier i programmet.
12. Annen tilhørende dokumentasjon til programsystemet organiseres i egen mappe og følger leveransen av programsystemet (dvs. det som finnes av: kravspesifikasjon, designbeskrivelse (wireline, pseudokode), testplaner, brukerdokumentasjon, installasjonsveiledning).