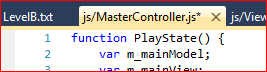
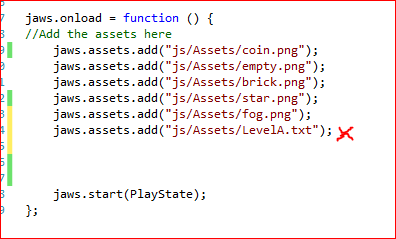
# Läsa in levels från textfil

Detta var svårare än man kunde trott. Det gick inte att placera filen i jaws.asets som man använder för att läsa in bilder

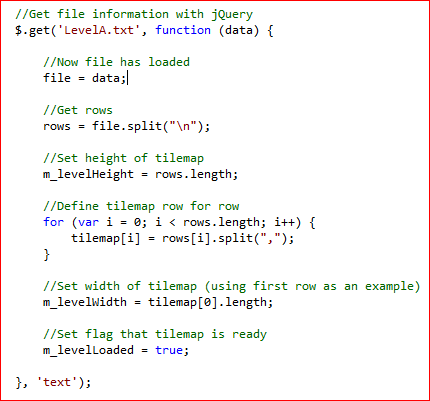


…

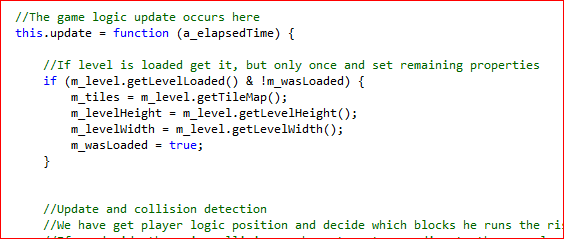


Jag fick därför ladda in jQuery och med hjälp av $.get metod (som är en shortcut av $.ajax) hämta ut texten.

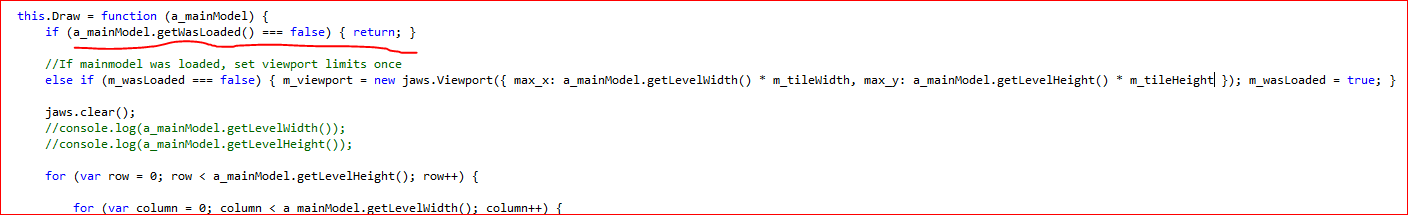




Inne i function(data) har hämtningen lyckats och vi kan kan skapa en tilemap utifrån vad som finns där. Problemet är ju dock att MainModel (som ju skapar Level) behöver skapa ett antal egenskaper utifrån hur många rader och kolumner tilemap i Level är, och detta vet vi ju inte direkt. Detta fick jag lösa genom att i Updatemetoden (som ju uppdateras en gång var sextionde sekund eller något) kolla om Level hade laddat klart, vilket jag i Level indikerade med m\_levelLoaded=true ovan. När Level hade laddat klart sattes de egenskaper i MainModel som var beroende av att Tilemap laddat klart.



Samtidigt var jag också tvungen att sätta en flagga i MainModel (kallad m\_wasLoaded) som gjorde dels att vi inte satte värdena på variblerna mer än en gång (jobbigt att sätta samma värde 60 ggr per sekund), men också så att MainView.Draw(), som ju får MainModel som argument, inte börja söka efter dessa variabler i MainModel som ännu inte satts.



I mainView används också en flagga, m\_wasLoaded, för att inte sätta maxvärden på viewporten mer än en gång (maxvärdena, x och y, på viewport kan bara läsas ut då Level läst in texten och avgjort hur många rader och kolumner vår tilemap har. Man skulle kunna tänkt sig också att MasterController läste in textfilen, och först när det var gjort så initierades instanser av MainModel (och därmed Level) samt MainView. Den hade dock antingen a) behövt göra tilemap global, eller behövt skicka med den när MainModel skapas vilket verkar lite bökigt. Kanske c att det hade gått att lägga den i jaws.assets, och komma åt den därifrån, det vet jag inte. Hade det gått hade det nog varit det bästa.

# Om ljudfiler

Minst ett problem uppstod, och det var att ett och samma ljud måste spela färdigt innan det kunde spelas igen. Och man kunde inte ha olika medlemsvariabler som pekade på samma ljudresurs, för då kraschade programmet. Man var istället tvungenn att ha en massa olika varianter av samma ljud kopplade till olika medlemsvariabler, ja redan då man laddar in ljuden så måste man ladda in flerra versioner av samma ljud. I mitt fall behövde jag (minst) 11 olika varianter av myntljudet i MasterController, och sedan koppla dessa till 11 olika variabler i MainView. Och detta trots att jag kortade ned det väsentligt, vilket gör att samma ljud kan börja spela tidigare.