**Test (test.js)**

Gemensam modul för samtliga funktioner. Denna anropas direkt från main.js (Test.start). Start funktionen hämtar gällande test (getCurrentTest). Första gången kommer jag till en en funktion som ger en allmän intro till testet varifrån jag klickar vidare till showInstruction från vilken jag beroende på var jag i testet är presenterar info från objektet allTests.

Exempel:

["testWords",

"Ordkunskap",

'Välj rätt betydelse på ordet i frågan. Du får poäng för ' +

'varje rätt svar.<p>Klicka för att starta testet...</p>',

],

Funktionen dispatcher är central och dit kommer jag återkommande för att avgöra vilket mitt nästa steg är. Från denna funktion kommer jag till de olika test-modulerna.

Några viktiga centrala funktioner som ligger i denna modul är:

Funktionen reset: anropas med Test.reset() för att fuska och börja om på aktuellt test.

Funktionen createQuestion: här skapas aktuell fråga som skrivs ut. I denna funktion anropas även check1 för att kontrollera svar mot facit.

Funktionen points: i denna kan jag både öka lokala och global poäng samt göra rest. Loka poäng för det aktuella testet och globala för alla tester.

Funktionen showTotalpoints: presenterar alla mina ihopsamlade poäng.

Funktion showFinalpoints: snabblösning i slutet för att visa den slutliga poängen. Denna hade jag kunna baka in i showTotalpoints.

**Deltest 1: Ordkunskap**

I modulen words.js skapade jag en array av objekt där varje fråga är ett eget objekt med egenskaperna 'question' (sträng: frågan), 'options' (array: svarsalternativ), 'answer' (sträng: det rätta svaret).

Exempel:

var ordkunskap = [{

question: 'Vad betyder "Klimakterium"?',

options: ["befruktning", "höjdpunkt", "övergångsålder"],

correct: "övergångsålder"

},

Jag använder funktionen nextQuestion för att läsa från objektet ordkunskap. Så länge det finns en ny fråga fortsätter testet. Jag använder några centrala variabler i samtliga test-moduler för att bla hålla reda på vilket test jag är på samt vilken fråga i det aktuella testet jag befinner mig på. Själva frågan skickas till Test-modulen där jag har funktioner som används av samtliga moduler.

**Deltest2: FizzBuzz**

Detta test liknar föregående test med den skillnaden att denna endast har en fråga. Här visas FizzBuzz-sekvensen där fortsättningen ska gissas på.

**Deltest3: Memory**

I denna visas 6 flaggor under en kort stund som sedan mha av byts ut mot tomma flaggor där det gäller att komma ihåg var respektive flagga initialt positionerades.

Jag har en initfunktion där jag skapar ett objekt för varje flagga. För varje flagg-objekt anropas därefter initfunktionen där jag populerar två arrayer med objekten. Flags[], här lägger jag flaggorna som orginalet och FlagsCopy[] använder jag till att slumpa ut flaggorna på.

Med setTimeout ( i startfunktionen) bestämmer jag hur länge flaggorna ska visas. Därefter anropas flip-funktionen i vilken läser från de utskrivna flaggorna, hämtar ut aktuella attribut och ändrar dessa till attributet: flag empty. Mha av css ändrar jag därmed utseende på dessa. Jag anropar funktionen randomFl, en viktig funktion som läser FlagsCopy arrayen och slumpar ut och håller reda på samtliga flaggor som jag har att skriva ut och returnera härifrån flagobjekt samt attribut som behövs.

**Deltest4: Shapes**

I denna har jag skapat objektet htmlShapes som innehåller namn, class och descr på de olika formerna som jag sedan vill skriva ut.

Exempel:

var htmlShapes = [{

name: "blueTriangle",

class: 'triangle blue shapeTile',

descr: "Den blå triangeln",

},

Från startfunktionen anropas:

progress(); Denna för att hålla reda på tiden jag har att svara på, i detta fall 10 sekunder.

drawDescription(); denna skriver ut från objektet htmlShapes.descr beskrivningen på figuren. Dessa slumpas ut och meningen är att man under 10 sekunder ska välja rätt figur som motsvarar beskrivningen.

drawShapes(); Denna skriver ut alla figurer utifrån objektet htmlshapes där name och class används i css för att bestämma utseendet.

Varje gång jag klickar på en figur ska är det tänkt att jag ska anropa check-funktionen där svar mot facit checkas och lokala poäng räknas upp.

När tiden har gått ut anropas showFinalPoints(); som är den sista anhalten där alla mina poäng visas.

**Kmom10**

Grundprincipen jag har följt är en modul per sektion (d.v.s. inledning, 4 deltester och avslutning), med externt gränssnitt enligt: 'name' (rubrik för sektionen), 'init' (initiering), 'reset' (omstart), 'points' (poäng för rätt svar), 'maxPoints' (maximalt möjlig poäng). Några har fler, andra färre.

Utöver dessa gjorde jag en separat modul, Shapes, för att hantera färg- och form-objekten. Den innehåller funktionalitet för att skapa och hålla reda på kombinationer med tillhörande objekt, svara på frågor om vilka kombinationer som använts och generera HTML-element för dem. Dess publika gränssnitt låter oss initiera modulen, skapa ny unik kombination, fråga efter använda kombinationer (namn eller objekt) eller skapa HTML-element för namngiven kombination. Namn på en kombination är färg och form, t.ex. 'red square', vilket matchar de CSS-klasser objekten får. Tillgängliga färger och former lagras inne i modulen och informationen är inte tillgänglig utanför.

Gemensam funktionalitet ligger i funktioner direkt under huvudmodulen (Test) - det är generiska funktioner som slumpmässig array-sortering samt HTML-specifika funktioner som t.ex. borttagning av HTML-element och funktioner som är specifika för den här uppgiften, som t.ex. skapande av länk till nästa fråga/sektion. I huvudmodulen ligger även variabler för poängräkning, en lista över vilka sektioner programmet ska gå igenom samt vilken av sektionerna som just nu är aktuell.

Varje sektion startas genom att dess init-funktion anropas, men därefter ansvarar modulen själv för vidare exekvering. Exakt när och hur det görs skiljer sig mellan modulerna, men gemensamt är att de använder funktionen som skapar nästa-länk för att användaren ska kunna gå vidare.

**Deltest 1: Tipsfrågor**

Här definierade jag frågorna som en array av objekt där varje fråga är ett eget objekt med egenskaperna 'question' (sträng: själva frågan), 'options' (array: svarsalternativ), 'answer' (sträng: det rätta svaret) och 'style' (sträng: css-klass för svarsalternativen/knapparna).

Exempel:

{

question: 'Vilken färg har mjölk?',

options: ['röd', 'blå', 'vit'],

answer: 'vit',

style: 'tips' // lägger '1.', 'X.' eller '2.' före texten

}

Deltesten har det publika gränssnitt som beskrevs ovan (name, init, reset, points, maxPoints), samt 'nextQuestion' som returnerar objektet för nästa fråga - eller undefined om inga fler finns.

Till detta skrev jag ett antal olika hjälpfunktioner (som fick hamna utanför modulen för att kunna användas även av andra deltester) för att skapa den struktur av HTML-element som behövs för att ställa frågan till användaren via webbsidan och därefter ta hand om användarens klickade svar.

I princip är det som genereras för varje fråga ett textstycke med frågan som sådan samt en knapp för respektive svarsalternativ. Knappar valde jag för att använda något annat än text och det är praktiskt att de går att deaktivera och även är ganska lätta att ändra utseende på. Ordningen på svarsalternativen slumpas fram för att ge variation. Knapparna får stilar som gör att de ser ut som text och varje knapp får en eventlyssnare som vid klick anropar en "kontrollera svaret"-funktion.

Denna funktion tar bort alla eventlyssnare (så att man inte kan klicka flera gånger), jämför klickat svar med det rätta (vilket skickas med i funktionsanropet) och lägger till en lämplig CSS-klass på det alternativ som klickats. Om svaret var fel läggs även en CSS-klass på det rätta svaret. CSS:en gör att rätt svar ringas in i grönt och fel svar i rött - kanske inte idealiskt för rödgrönt färgblinda.

Länk till nästa fråga (eller nästa sektion, om frågorna är slut) skapas/visas när svaret kontrolleras.

**Deltest 2: Fizzbuzz**

Denna deltest är väldigt lik föregående. Den huvudsakliga skilladen är att den här bara har en fråga, och att den frågan innehåller mer än bara text - här ingår också FizzBuzz-sekvensen som användaren ska gissa fortsättningen på. Därav är 'question' en struktur av HTML-element snarare än bara ett. Koden i denna modul, liksom i deltest 1, är tämligen kortfattad och eftersom det bara finns en fråga har modulen heller ingen 'nextQuestion' i sitt externa gränssnitt. CSS-klassen som används för knapparna är 'button', vilket stipulerar att texten ska föregås av en räknare och då har jag valt latinska gemener, d.v.s. svarsalternativen föregås av "a.", "b.", "c." etc.

**Deltest 3: Bildminne**

Här fick jag tänka annorlunda eftersom testen är upplagd på annat sätt än de två tidigare.

Eftersom vi här ska visa en förklarande text innan testen börjar skapade jag en ny gemensam funktion för att visa introduktionstexter - den tar emot en lista med texter samt en callback-funktion att anropa när användaren klickar på "gå vidare"-länken. Funktionen skapar (och visar) ett textstycke för varje bifogad text samt den länk som drar igång den egentliga testen, vilket då motsvarar det som i de tidigare deltesterna görs direkt i 'init'-funktionen. En sorts insticks-intro.

Starta-funktionen bygger upp en HTML-tabell om 3 rader om 3 celler vardera och lägger in en flagga i varje cell. Jag använde samma flaggor som i tidigare kursmoment och skapade några fler för att få ihop 9 unika flaggor. Namnen på de länder dessa flaggor tillhör slumpsorteras och sparas som en array på tabell-elementet, liksom också den ordning i vilken användaren senare ska få peka ut flaggorna. Varje tabellcell får dessutom ett id, som anger dess index i arrayen med namn.

Efter 5 sekunder byts flaggorna ut mot frågetecken och varje tabellcell får en eventlyssnare som tar emot klick. Den första eftersökta flaggan visas och användaren ombeds klicka på dess tidigare plats. När användaren klickat på en cell är det enkelt att med dess id (som anger relativ position) hitta namnet i flagglistan och jämföra det med namnet på den efterfrågade flaggan.

Eftersom deltest 3 ska handla om bildminne och inte om att testpersonen ska veta vilka flaggor som hör till vilka länder avviker jag från kundens specifikation och visar flaggor i stället för en lista med landsnamn. Eftersom flaggorna skulle göra sig dåligt i en lista som dessutom skulle ta väldigt mycket plats på höjden, visar jag i stället en flagga åt gången. När användaren klickar på en cell visas flaggan som låg i den. Om det var fel visas även den rätta flaggan, men något transparent.

Det externa gränssnittet innehåller endast name, init, reset och maxPoints.

Det här är den deltest jag själv tycker är svårast, så egentligen skulle jag vilja ge 5 poäng för varje rätt klickad flagga men valde ändå att hålla mig till specifikationen och bara ge 1 poäng.

**Deltest 4: Visuell förmåga och läsförståelse**

Ytterligare skillnader jämfört med tidigare deltester gjorde att det även här blev mycket tänkande och kodande. Inledande förklaringstext hanteras enkelt med den gemensamma funktion för ändamålet som skapades för föregående deltest och därefter skapas 10 objekt med hjälp av Shapes-modulen. Objekten visas i webbsidan tillsammans med instruktionen och en lista med objektbeskrivningar - samtidigt som klockan börjar ticka och räkna ned de 15 sekunderna.

Varje gång användaren klickar på ett objekt markeras aktuell beskrivning i listan som förbrukad genom att texten ändrar färg till ljusgrå - detta gör det enkelt för användaren att se vilken beskrivning som gäller. När alla klick är förbrukade (eller tiden gått ut) markeras de rätt klickade beskrivningarna så att de blir gröna och får en efterföljande grön bock. Detta görs genom att lägga till en CSS-klass på elementet. Jag anser att detta visuella motsvarar att visa resultatet.

En rolig utmaning i den här deltesten var att koppla ihop tekniska egenskaper och textuella beskrivningar. Jag löste det med hjälp av en ordlista, där jag utgick från listan med använda färg- och formkombinationer. Denna lista (bestående av 'färg form'-texter) slumpsorteras och översätts därefter till hela uttryck genom att översätta färg och form samt slå ihop dem med "Den " och ".". Ja, och så blir varje uttryck ett eget listelement i en ordnad HTML-lista, som visas i webbsidan.

Ordlistan ser ut så här:

var dictionary = {

red : 'röda',

white : 'vita',

black : 'svarta',

blue : 'blå',

green : 'gröna',

yellow : 'gula',

orange : 'orange',

circle : ' cirkeln',

square : 'kvadraten',

triangle : 'triangeln',

rectangle : 'rektangeln'

};

Även i den här deltesten utnyttjar jag det faktum att man kan lägga till information på objekt. Här får varje listelement en egenskap som heter colorShape, som motsvarar namngivningen (t.ex. 'green triangle'). När användaren klickar på ett objekt, lagras det i en lista. När testen avslutats är det enkelt att löpa igenom de förbrukade listelementen och med hjälp av colorShape kontrollera om motsvarande klickat element har önskad färg och form, d.v.s. de önskade CSS-klasserna.

Här hade jag också roligt med att skriva koden för gruppering av objekten, för det kändes inte så värst användarvänligt att lägga dem i en enda lång rad. Det vore särskilt besvärligt om fönstret var brett. Ett tag tänkte jag bowling och fördelade objekten 1, 2, 3, 4 men sedan tyckte jag det var trevligare att hålla dem så samlade som möjligt så jag fastnade för fördelningen 3, 4, 3.

Det externa gränssnittet innehåller endast name, init, reset och maxPoints.

**Deltest 5: Uppfattningsförmåga**

Här visas också en inledande förklaringstext, som länkar vidare till start av testen. Det slumpas fram vilket av specifikationens tre kriterier som ska användas. Kriterierna ligger i en objektlista, där varje objekt har en textbeskrivning och en funktion för matchning. Kriteriets textbeskrivning skrivs ut, objekten hämtas (återanvänds från deltest 4) och utfrågningen tar sin början.

Efter en sekund visas nästa objekt, efter ytterligare en sekund tas detta bort igen. Oavsett om användaren klickar på objektet eller ej matchas det mot aktuellt kriterium. Om användaren klickat och objektet matchar utdelas poäng, liksom om användaren inte klickat och objektet inte matchar kriteriet. Man får alltså poäng för att göra rätt, oavsett om det innebär att klicka eller att låta bli.

När testen är avslutad visas en text med antalet insamlade poäng, då det annars inte finns något självklart sätt att visa resultatet - det finns ju inga element att lägga till CSS-klasser på.

Det externa gränssnittet innehåller även här endast name, init, reset och maxPoints.

**Om projektet**

Det här var ett roligt, utmanande och intressant kursmoment! Här gavs möjligheter att använda kunskaper och erfarenheter från tidigare kursmoment, men jag måste erkänna att jag tidigt låste mig vid att använda moduler för de olika deltesterna. Så här i efterhand borde jag kanske ha beaktat andra tekniker. Jag testade med riktiga klasser, men valideringen gillade visst inte det.

Rent allmänt är jag inte helt nöjd med min lösning - jag snöade in på det där med en modul per deltest och sedan blev det för jobbigt att testa andra varianter. Detta innebär bl.a. att koden innehåller en del upprepning och åtminstone för mig känns spretig. Jag hade gärna velat jobba mer med prototyper och att generalisera mina funktioner, men ork och fokus tog helt enkelt slut. Jag tröstar mig dock med att testerna ändå fungerar och att jag (när jag väl får det riktiga uppdraget!) kommer att ha alla möjligheter implementera bättre och snyggare lösningar.

Det är med JSDoc var ju ett kapitel för sig. Koden blir väldigt mycket längre med kommentarerna! Då jag jobbar med Java till vardags är jag van vid JavaDoc så strukturen är mig inte främmande. Jag installerade jsdoc med en tanke om att kunna verifiera min dokumentation, men fick det inte att fungera direkt och ville inte avsätta en massa tid för felsökning. Det är dock alltid nyttigt att dokumentera sin kod eftersom det kräver att man både förstår och tänker igenom den, vilket påfallande ofta leder till ändringar och förbättringar av olika slag. Så också här och nu.

Projektet som helhet var roligt för att det erbjöd många små utmaningar eftersom deltesterna i viss mån liknande varandra, men ändå alltid på någon avgörande punkt var olika. Detta gjorde mitt ordningssinne lite obekvämt då jag gärna strävar efter konsekvens och likformighet, men det blev också en nyttig personlig utmaning att tillåta mig att skriva mindre homogen kod.

Det gick bra att genomföra projektet och jag stötte inte på några särskilda problem vad jag minns, annat än att jag fortfarande tycker att det är knepigt med hur man bör tänka när man jobbar med/mot webbsidor. Det blir lätt lite kladdigt kodat, med logik och presentation ihop där jag egentligen skulle vilja separera dem. Jag antar dock att sådant ger sig när man blir mer van, när man fått chansen att prova många olika tänkbara sätt att lösa de problem man ställs inför.

Omfattning och svårighetsgrad på projektet var alldeles utmärkt och i mitt tycke precis lagom. En del utmaningar, men inga som kändes riktigt jobbiga utan mer inspirerande och stimulerande.

**Om kursen**

Jag är mycket nöjd med kursen, tycker att jag har lärt mig massor. Materialet har varit adekvat och jag tycker framförallt att det varit nyttigt att tvingas bekanta sig med MDN. Det var tufft i början, men det är en väldigt bra resurs att kunna använda. Jag har (till min förvåning) varmt uppskattat videorna med Douglas Crockford - skulle möjligen önska att de var aningens färskare. Vad gäller handledningen kan jag inte säga så mycket då jag knappt utnyttjat den denna termin.

En mindre detalj gällande kursmaterialet: webbsidan för kursmomentet har ett duplicerat id, 'tips' (används även för Deltest 3, Minne). Jag testade även valideringen hos W3C - den höll med. :)

Jag har och kommer att rekommendera kursen till andra. Mitt betyg: 9 av 10.