Generella dataformer

* Skrivs som ren text (jämfört med bilder, ljud mm, dessa kallas för binärfiler)
* Har en bestämd syntax, precis som kod, som beskriver innehållet.
* Har inget bestämt användningsområde, men används bland annat för att lagra eller skicka data.
* **Markup filer GÖR aldrig någonting**
* **Exempel: XML, JSON, YAML, MARKDOWN**
* HTML, CSS? Det är inga generella dataformer

Metadata är tabbar </> eller ””. Data är det som ligger mellan taggar. Eller den information som kommer in.

* Varför? För att rena textfiler blir svårlästa
* För att införa metadata’
* Enkelt att läsa av både människor och datorer
* Ett bra mellanalternativ för mindre mängd information, innan man behöver en databas.
* Varför inte? Vid för stor mängd blir det oftast klumpigt (som med stora hemsidor)
* Databaser är snabbar
* Binärfiler (mp3, jpg ect) är mycket snabbare att \*parsa\* för en dator

**XML**

XML, är inte lika vanligt längre. Det är mycket extra karaktärer och filen blir större. Det är mycket information.

* Står för eXtensible Markup Language
* Mest känd för SGML (Standard Generalized Markup Language) formaten, tillsammans med HTML
* Det finns inga förbestämda taggar (som p, div, ect)
* Taggar kan definieras i ett XML-schema
* Det finns många färdiga parsers
* XML kan ses som ett SGML språk som passar allt, medans HTML ”är” ett SGML språk för hemsidor
* <TAG attrubut=”värde”>innehåll</TAG>
* <SjälvstängandeTag/>
* <Parent><Child></CHILD></Parent>
* <METADATA>DATA</METADATA>
* Spelar en viktig roll i många olika IT-system
* Används ofta för att distribuera data över Internet
* Det är viktigt för utvecklare att ha en god förståelse för XML

**JSON**

* JavaScript Object Notation
* Uppfanns för att XML innehåller för mycket onödig information för att skicka fram och tillbaka mellan en server och klient
* Nästan likadan syntax som JS objekt, men JSON är inte bundet till JavaScript
* Parser inbygd i JavaScript.
* Kspm består av ett { kay: value… } objekt (firstname, lastname ect)
* Key är alltid en stäng
* Value är en sträng, lista, number, boolean, ett objekt eller null
* { Key: value }
* { METADATA: DATA }

**Storage**

* Med **sessionStorage** och **localStorage** kan vi spara information i våra besökares webbläsare
* Värden sparas i sessionStorage sparas till användaren stänger ner fliken/webbläsaren. Det är bara temporärt.
* Värden sparade i localStorage sparas till användaren tömmer sin cache. Tex en varukorg. Dina varor eller ”likes” ligger kvar i kundkorgen tills att man tar bort dom manuellt.
* Är ett objekt av strängvärden
* Är read-only vilket innebär att man vi måste ersätta värdet helt om vi vill förändra det.
* Support i alla moderna webbläsrae
* **OBS!** Det räcker därmed inte att ladda om sidan för att ta bort data du lagt till, dessa måste tas bort manuellt eller via din kod!

**Storage vs Cookies**

* localStorage och sessionStorage sparas enbart client side, alltså, vår server side har inte tillgång till innehållet.
* Skapar man coockies däremot skickas den informationen från client side (dvs webbläsaren) till srver sida (dvs servern som hanterar webbsidan) vid anrop. (Mer om cookies i framtida kurs)
* Storage kan hantera mycket störra mängder data.
* Spara sånt i client side som berör användaren/besökaren, med tex användarinformation, inställningar eller inmatad data i formulär
* Spara inte sådans som är kritiskt och inte går att ”återfå” för användarens kan tömma alla sina storage och cookies.
* Näs genom window.localStorage
* Är ett objekt
* Är read-only
* Har inga expirationsdatum, vilket skiljer den från tex sessionStorage
* OBS! Det räcker därmed inte att ladda om sidan för att ta bort data du lagt till, dessa måste tas bort manuellt.
* Nås genom window.localStorage eller direkt genom localStorage
* localStorage.setItem, localStorage.removeItem
* Hur gör vi om vi vill spara något annat värde än en sträng?
* JSON.Stringify för att spara
* JSON.Parse för att läsa

**Vad JSDOC**

* JSDoc är ett märkspråk som används för att dokumentera källkod i JavaScript med hjälp av kommentarer.
* Med beskrivande text i dina JSDoc-kommentarer skapas dokumentationen för din applikation som beskriver hur funktioner och andra koddelar är tänkta att användas.
* Det går också att definiera typer så du som utvecklare får intellisense när du skriver kod i tex VSCode.
* JsDoc kommentarerna kan sedan bearbetas för att producera dokumentaionen i exempelvis HTML
* @param – används för att beskriva argumenten till funktioner

@return – används för att beskriva vad funktionen retunerar

@typedef – används för att definiera en ny typ, tex ett object och dess innehåll