Frontend-programmering: hur har webbsidans design utvecklats i samband med frontend-programmeringens utveckling

Hilda Malm 2001153
Kandidatavhandling i Datateknik
Handledare: Annamari Soini
Fakulteten för Naturvetenskaper och Teknik
Åbo Akademi
2023

Referat

De allra första hemsidorna bestod av text och blåa, klickbara länkar. De byggdes upp av tre protokoll: HTML, HTTP och URL. Alla webbsidorna såg likadana ut på grund av att ingen stilmall fanns tillgänglig för att ändra utseendet på webbsidan.

Idag är nästan alla webbsidor unika. Det finns CSS som hjälper ägaren att personalisera sin webbsida med färger, former och layout. Det finns JavaScript som möjliggör förändringar i webbsidans beteende och gör det trevligare för användaren att använda webbsidan. Det finns idag många ramverk för flera olika syften, bland dem är React och Angular baserade på JavaScript. De används i stor omfattning i dagens webbutveckling till följd av att de förenklar programmeringsprocessen och effektiviserar utvecklarnas arbete.

Syftet i den här avhandlingen är att analysera frontendprogrammeringens utveckling och nya programmeringsspråk och ramverk som används i frontenddesign idag. Webbsidor ska idag vara snabba, interaktiva, flexibla och tillgängliga för så många som möjligt, och en webbutvecklare ska idag ha mycket mera kunskap inom flera programmeringsspråk, ramverk och verktyg. Det finns mängder med information på internet i form av forum med problemlösningar, artificiella intelligenser som kan lösa uppgifter och videor med förklaringar på otaliga mängder koncept. Frågan är slutligen om den stora mängden information är till en utvecklares fördel eller nackdel.

Innehåll

Terminologi

- W3C - World Wide Webb Consortium - är en internationell gemenskap som arbetar för att utveckla standarder för World Wide Webb.
- W3C Recommendation - en specifikation eller flera riktlinjer som fått stöd av W3C medlemmar och direktören Tim Berners-Lee.
- Netscape Navigator - en sökmotor som utvecklades av Netscape och stängdes ner år 2008.
- IETF, Internet Engineering Task Force - en organisation som ansvarar för tekniska standarder för Internetprotokoll.
- RFC - *Recuest For Comments*, är ett formellt dokument från IETF med specifikationer angående ämnen relaterade till internet och datornätverk. Det finns olika dokument av RFC med olika specifikationer. RFC 1945 behandlar HTTP/1.0.

Kapitel 1

Introduktion

I den här avhandlingen analyseras utvecklingen av frontend-programmering med ett fokus på hur design påverkats av programmeringsspråkens utveckling. Avhandlingens syfte är att ge människor som är intresserade av frontend-programmering, webbdesign och webbutveckling en inblick i webbens historia och evolution, samt faktorer som bidragit till och accelererat förändringar i webbsidor.

Genom att analysera hur frontend-programmering och webbsidors design utvecklats kan man dra slutsatser angående faktorer som bidragit till utvecklingen och nya verktyg och koncept inom webbutveckling. Med hjälp av undersökningar av artiklar och bloggar som behandlar programmeringsspråkens historia och evolution, trender i design, samt bloggar med förstahandserfarenhet från webbutvecklare som arbetat med webbsidor sedan tidigt 90-tal har avhandlingen nått sitt resultat.

Varje webbsida har sin unika design, men i grunden följer de ett fåtal stilprinciper som bestämmer hur en webbsida ska se ut. Generellt sett ska en webbsida ha ett navigationsfält, en logo eller en text för företaget eller privatpersonen, och ett något strukturerat upplägg. Principerna och trenderna för hur upplägget eller navigationsfältet ser ut har hunnit ändra flera gånger sedan tidigt 90-tal. I den här avhandlingen nämns några av dessa förändringarna, såväl som hur programmeringsspråk och tillägg har understött dessa förändringar.

Frontend-programmeringen idag består inte bara av kodning. En webbutvecklare ska idag ha mera kunskap inom flera områden än det behövdes i början på 2000-talet. Idag ska en utvecklare veta vilka ramverk och bibliotek som finns tillgängliga, vilka som fungerar till olika typer av projekt och även kunna tillämpa dem. Informationsmängden på webbsidor har ökat sedan tidigt 2000-tal, vilket tvingar utvecklare och frontend-programmerare att tänka om och strukturera webbsidan så att all information ryms med och dataflödet inte påverkar webbsidans användbarhet. Dessutom ska navigationen i webbsidan förbli intuitiv och tydlig.

Kapitel 2

Webbsidan: historik och beståndsdelar

En webbsida är ett HTML-dokument som är ihopkopplat med andra HTML-dokument via hypertextlänkar. Den första webbsidan, World Wide Web, bestod av HTTP (eng: *Hypertext Transfer Protocol*), HTML (eng: *Hypertext Markup Language*) och URL (eng: *Universal Resource Locator*) och alla tre protokollen utvecklades av Tim Berners-Lee. Idag är webbsidor mångsidigare och kan personaliseras med hjälp av CSS (eng: *Cascading Style Sheets*) och JavaScript, samt diverse ramverk och bibliotek.

2.1 Hypertext Markup Language, HTML

2.1.1 Vad är HTML?

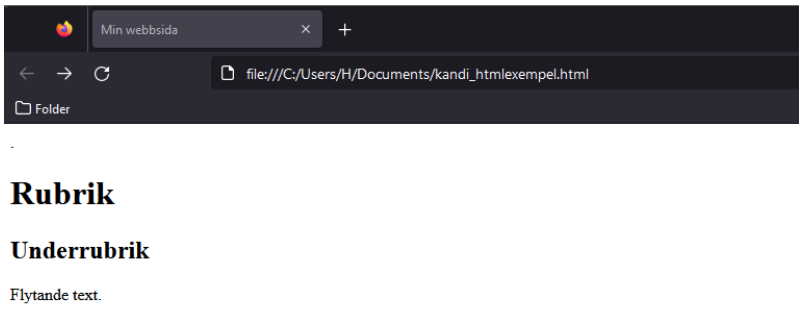
HTML (eng: *Hypertext Markup Language*) är inte ett programmeringsspråk, utan ett märkningsspråk som innehåller länkar som gör det möjligt att hoppa mellan delar av ett dokument eller ut ur ett dokument in i ett annat. HTML används för att skapa hypertextdokument som är plattformsoberoende och alla webbsidor är uppbyggda på det. [10]

HTML-koden i listing 2.1. består av taggarna (eng: *tags*) <head>, <title>, <meta>, <body>, <h1>, <h2> och <p>. De flesta taggar har en öppnande <tag> och en stängande </tag>. Se figur 2.1 angående hur HTML-koden i listing 2.1 ser ut i Mozilla Firefox. Taggen <head> inkluderar alltid en <title>, i vilken man kan ändra namnet som visas i webbläsarfönstret. Taggen <meta> lagrar information om dokumentet, till exempel karaktärkodning, namn och beskrivning. [24]

```
1 <html>
2     <head>
3         <title>Min webbsida</title>
4         <meta charset="UTF-8">
5         <meta name="beskrivning" content="Beskrivning av din
    hemsida">
```

```
    </head>
    <body>
        <h1> Rubrik </h1>
        <h2> Underrubrik </h2>
        <p> Flytande text. </p>
    </body>
</html>
```

Listing 2.1: Exempelkod i HTML.



Figur 2.1: Resultat av HTML-koden från listing 2.1.

2.1.2 HTMLs historia

HTML började utvecklas år 1989 och den första versionen, HTML 1.0, publicerades år 1990 och användes mellan år 1990 och 1994. Den innehöll endast 20 element, vilket begränsade antalet justeringar man kunde göra och slutresultatet blev att majoriteten av webbsidorna såg likadana ut. HTML 1.0 fanns före W3C (eng: *World Wide Web Consortium*) grundats, vilket innebär att det inte finns mycket detaljer angående den första versionen. Det man vet är att HTML 1.0 stödde grundläggande elemet som text och bilder, men hade ingen implementation för tabeller eller fonter. År 1995 började HTML 2.0 användas och blev den första standarden för HTML. HTML 2.0 var ämnat att förbättra den första versionen, vilket inkluderade att tabeller implementerades som en tagg, fonter och färger kunde justeras, och vissa webbläsare började implementera webbläsar-specifika taggar. W3C hjälpte utvecklingen av HTML genom att fastslå standarder som skulle gälla över flera webbläsare så att de alla uppfattade och laddade in HTML-taggar på liknande sätt. HTML 3.2 togs i bruk 1997 och nu kunde man till exempel ha text runt bilder, den stödde CSS (eng: *Cascading Style Sheets*) och *frame*-taggar. *Frame*-taggar används för att dela en webbsida i delar som sedan kan laddas skilt. Tack vare CSS såg HTML-taggar bättre ut i webbläsare. År 1999 publicerades HTML 4.01 och nu utökades stödet för CSS och nya HTML-taggar. CSS-koden behövde inte längde vara integrerat i HTML-dokumentet utan kunde vara i ett helt separat dokument. HTML 5 är den nuvarande versionen av HTML

och fungerar på alla de största webbläsarna, till exempel Firefox, Chrome, Safari och Internet Explorer. Bland annat lades ett input-element av typen email till som kontrollerar att det insatta värdet är en giltig email, en tagg för lösenord som inte visade vad användaren skriver utan istället visas specialsymboler, och ljudtaggar lades till för att ge möjligheten att lägga ljud till i sin webbsida. Semantiska taggar, även kallade strukturella taggar, var också en ny implementation. Den här taggen hjälper att dela in HTML-sidan i olika strukturer och bland annat är <header> och <footer> semantiska taggar. En section-tagg används för att dela in HTML-sidan i olika sektioner. HTML5 gjorde att man kunde skriva webbsidor utan att vara uppkopplad till internet, den gjorde det möjligt för webbsidor att veta var man befinner sig, och den klarade av att spela upp videor med hög upplösning och leverera bättre grafik. [25][18]

2.2 Cascading Style Sheets, CSS

2.2.1 Vad är CSS?

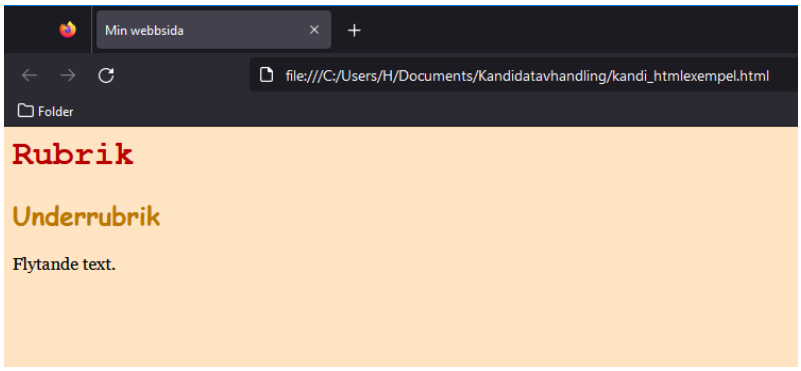
CSS (eng: *Cascading Style Sheets*) används i frontend-programmering för att lägga till stil i webbdokument genom att ändra till exempel font, färg och mellanrum. CSS-formatmallen gjorde det så att man hade dokumentet i en fil och sitt stilark i en annan. Det gör att man lätt kunde ändra stilen utan att tänka på innehållet. I webbläsaren omvandlas HTML till en DOM (eng:*Document Object Model*), vilket innebär att den översätts till en trädstruktur där varje nod motsvarar ett objekt i dokumentet. CSS går igenom DOM-trädet för att nå HTML-taggarna. Till exempel når den först <html>-taggen och inuti <html>-taggen finns <head>-elementet, etc. [6]

CSS kan läggas till ett HTML-dokument på tre olika sätt: inbakad stil (eng: *inline styling*), internt stilark (eng: *internal style sheet*) eller externt stilark (eng: *external style sheet*. När CSS-koden skrivs skillt i varje tag eller komponent i HTML-dokumentet kallas det för inbakad stil. Inbakad stil blir onödigt duplicerad vid större projekt där en stil ska användas på flera olika element. Om all CSS-kod finns i <head>-taggen i HTML-dokumentet kallas det för internt stilark. CSS-koden appliceras då till alla element i <body>-taggen, vilket inte är användbart för flersidiga webbsidor eller applikationer. Externt stilark används i större projekt eftersom CSS-koden finns i ett helt skilt dokument och kan kopplas till flera HTML-dokument utan att påverka HTML-strukturen. Det gör det lättare att avlusa (eng: *debug*) koden och spara på sidans laddningstid. [28]

Stilen på HTML-dokumentet i listing 2.1 kan förändras med hjälp av CSS. Koden i listing 2.2 förändrar HTML-taggarnas färg och font, och i figur 2.2 representeras webbsidan när stilarket laggts till.

```css
body {
    background-color: bisque;
}
h1 {
    color: rgb(185, 0, 0);
    font-family: 'Courier New', Courier, monospace;
}
h2 {
    color: rgb(187, 118, 0);
    font-family: cursive;
}
p {
    color: rgb(0, 0, 0);
    font-family: Georgia, 'Times New Roman', Times, serif;
}
```

Listing 2.2: Exempelkod i CSS.



Figur 2.2: Webbsidan efter att man lagt CSS till exempelkoden till HTML-dokumentet från listing 2.2.

2.2.2 CSS historia

CSS påbörjades i CERN, , Europeiska Organisationen för Kärnforskning av Håkon Wium Lie och publicerades år 1994. Tanken att separera dokumentets design från dess struktur var inte helt ny, eftersom det hade varit en framtidsvision sedan början av 1990 då den första versionen av HTML publicerades. Bert Bos var den första som uttalade sig om den första versionen av CSS. Han arbetade på Argo, en webbläsare som prioriterade anpassningsmöjligheter, och slog sig samman med Wium Lie för att utveckla CSS. [6]

År 1994 hölls en konferens i Chicago där CSS presenterades. Följande år, 1995, deltog Wium Lie och Bos i en ny konferens och fick nu visa implementationer av samma version av CSS som presenterats året före. Representanter med författarperspektivet påstod att beslut om design borde ligga helt och hållet på författarens sida. Däremot tyckte Wium Lie

och Bos att användaren skulle ha sista ordet när det dök upp konflikter i designen. HTML ERB (eng: *HTML Editorial Review Board*) grundades i slutet av 1995. När Microsoft antydde att de skulle börja stöda CSS år 1996 var Netscape tvunget att följa samma riktning. Syftet med att Netscape också skulle anpassa sig till CSS var att man skulle undvika att tvinga webben i två olika riktningar som stöder olika specifikationer. Netscape var motvilligt inställd till CSS och försökte kringgå stilarket genom att implementera mindre delar av CSS internt, vilket innebar att CSS regler översattes till bitar av JavaScript och döptes till JSSS (eng: *JavaScript Style Sheets*). Netscape lät sina utvecklare koda i JSSS och försökte på det viset undvika CSS helt och hållet. Det hela misslyckades likväl och CSS fick leva vidare medan JSSS upphördes. Sist och slutligen publicerades CSS level 1 som en W3C (eng: *World Wide Web Consortium*) *Recommendation* i slutet av 1996, trots att den då inte stöddes av alla webbläsare. Den gav möjligheten att ändra fonter, typsnitt, mellanrum (eng: *padding*), marginaler, gränser, orientation (eng: *alignment*) och tabeller. CSS fick sitt eget team i W3C för att åtgärda de saker som CSS1 inte hade implementerat. CSS2 var en förbättring av CSS1 och tog bort oanvändbara funktioner. Samtidigt som den tog bort oanvändbara funktioner lade den till absolut, relativ och fixad positionering av element, den stödde olika typer av media och inkluderade nya egenskaper för fonter, som skuggor. Opera, ett norskt företag, var den tredje webbläsaren som stödde CSS. Webbläsaren tog lite utrymme, fungerade i mobiltelefoner, stödde majoriteten av funktionerna och utvecklarna hade tid att testa CSS-implementationen ordentligt. Wium Lie var så fascinerad av Operas teknologi att han gick med i företaget 1999. CSS3 är den senaste versionen av CSS och är standard i dagens frontend-programmering eftersom den stöds av nästan alla webbläsare. Den fokuserar på modularisering, stöder animationer och har tillagt möjligheten att runda hörn i rutor, skuggor för lådor och anpassa storleken på en låda enligt innehåll. [6][28]

2.3 Hypertext Transfer Protocol, HTTP

Tim Berners-Lee skrev och publicerade protokollet HTTP (eng: *Hypertext Transfer Protocol*). Det är ett av de tre grundläggande elementen för webben. HTTP är ett klient och server protokoll i tillämpningsskiktet. Det används för att hämta resurser, som till exempel HTML-dokument och bilder, med hjälp av efterfrågnings-metoder (eng: *request methods*). Det finns två typer av meddelanden i HTTP: efterfrågningar (eng: *request*) och svar (eng: *responses*). En HTTP-metod är antingen ett verb eller ett substantiv och alltid skrivet i versaler. De mest använda request-metoderna är POST och GET. Med POST-metoden skickar man data till en server och med en GET-metod begärs data från en specifik resurs. [13][8]

HTTP/0.9, även kallat *the one-line protocol* eftersom en efterfrågan bestod av en enda kodrad, var den första versionen av HTTP. Det var enkelt att använda eftersom den inkluderade endast en metod, GET, som följdes av en sökväg till resursen man ville hämta. Även svaret var enkelt och bestod endast av filen man hämtade, som var ett HTML-dokument eftersom den här versionen inte hade HTTP-rubriker (eng: *headers*) som skulle har gjort det möjligt att hämta andra typer av dokument. Ingen felkod eller status var inkluderat, utan om det uppstod ett fel skickades en specifik HTML-fil tillbaka som förklarade vad felet var. I HTTP/1.0 lades versionen av HTTP med på samma kodrad som GET-metoden när man gör en efterfrågan. En status som berättade om efterfrågan hade lyckats eller inte lades till i början av svaret. GET var fortfarande den enda metoden tillgänglig. HTTP-rubriker för efterfrågningar och svar introducerades i samband med HTTP/1.1 och gjorde det möjligt att överföra andra typer av filer med *Content-Type*-rubriken. Man kunde göra en efterfrågan efter ett HTML-dokument, och sen göra en efterfrågan efter en bild inuti dokumentet. Under åren mellan 1991 och 1995 försökte browsers och servrar att anpassa en ny funktionalitet och se om den tog fäste, men på grund av att interoperabilitetsproblem orsakade det mera irritation än glädje. I ett försök att lösa problemet publicerades ett dokument med information om allmänna tillämpningar för HTTP/1.0, *RFC 1945*. [1]

Den första standardiserade versionen av HTTP, HTTP/1.1, publicerades år 1997. HTTP/1.1 inkluderade flera förbättringar och förtydligade tidigare versioners oklarheter. Tidigare kopplingar kunde nu bli omanvända istället för att behöva öppna samma koppling flera gånger för att se innehållet i ett hämtat dokument. Kanalledning (eng: *pipelining*) var nu tillgängligt och tillät en andra efterfrågan att skickas före man fått svar på en tidigare, och på det sättet minskade kommunikationens fördröjning (eng: *latency*). Ihopklumpade svar och cache-kontrollmekanismer lades till. Innehållsförhandlingar (eng: *content negotiation*) togs i bruk för att hjälpa klienten och servern att komma fram till det mest lönsamma innehållet att utbyta. Detta inkluderade språk, kodning (eng: *encoding*) eller typ. *Host*-rubriken gjorde det möjligt att hålla olika domäner på samma IP-adress och *server colocation*. [1]

Idag innehåller webbsidor mycket mera innehåll och innehåll av flera olika media, vilket innebär att mera data överförs med flera HTTP-efterfrågningar. Google implementerade protokollet SPDY i början av 2010-talet som ett andra alternativ för att byta data mellan klienten och servrar. Den implementerade en förhöjning i mottaglighet (eng: *responsiveness*) och jobbade mot problemet med duplicering vid dataöverföring. SPDY lade grunden för HTTP/2, som blev standardiserat år 2015. HTTP/2 är ett binärt, multiplexerat protokoll med vilken parallella efterfrågningar kan köras över samma koppling, den komprimerar rubriker, och tillåter en server att lägga data i en klient-cache före klienten

frågat efter det med hjälp av en *server push*. Med hjälp av den nya versionen hade webbsidor med mycket trafik möjligheten att spara i dataöverföringskostnader och påföljande kostnader. HTTP/2 krävde ingen anpassning från webbsidor eller applikationer, utan det räckte med att ha en uppdaterad server och tillräckligt ny webbläsare. År 2016 publicerades flera tillägg till HTTP: *Alt-Svc* och *Client-Hints* lades till, och *Cookies*-rubriken fick nya prefix. *Alt-Svc* är en svarsrubrik (eng: *response header*) som visar vilka olika tjänster man kan använda för att nå en och samma resurs. *Client-Hints* gjorde det möjligt för webbläsaren att ge servern information om sina krav eller sitt hårdvarusystems begränsningar. Säkerhetsrelaterade prefix till *Cookie*-rubriken garanterar att en säkerhetskaka (eng: *secure cookie*) inte har modifierats. [2]

HTTP/2 är populärt idag eftersom det inkluderar riktad prioritering (eng: *weighted prioritization*) som låter utvecklarna att själv bestämma vilken del av webbsidan som laddas in först när man öppnar en webbsida. Beroende på vilken del av webbsidan som laddar först bygger användaren olika uppfattning om hur snabbt den laddar. Ifall navigationsfältet och rubriken högst uppe på sidan laddas först uppfattas webbsidan som att den laddar snabbare än om webbsidan laddas nerifrån och uppåt. Bortsett från att utvecklare kan påverka hur snabb inladdningen uppfattas så kan de bestämma vilken del av webbsidans byggnadsdelar laddas först. Större JavaScript-filer kan ta länge att ladda in och dess inladdning är inte synlig på webbsidan på samma sätt som HTML- eller CSS-filen ger en synlig förändring, så om JavaScript-filer prioriteras först kan det uppfattas som att webbsidan inte alls laddar. Med hjälp av multiplexering (eng: *multiplexing*) i HTTP/2 kan servern skicka flera strömmar med data till klienten på en gång istället för att skicka en del av data i gången. Varje dataström kan anges ett värde som, på klientens sida, berättar vilken ström som ska renderas först. I HTTP/1.1 stoppas hela dataströmmen efter en resurs som inte kan laddas. HTTP/2 använder en enda TCP-koppling för att sända flera dataströmmar samtidigt utan att någon resurs blockerar en annan. Detta gör HTTP/2 genom att den delar upp data i binära meddelanden och numrera dem, så att klientsidan vet vilken ström varje binärt meddelande hör till, och om en del data inte kan laddas syns det i numreringen. Både HTTP/1.1 och HTTP/2 använder rubrikkomprimering (eng: *header compression*), men HTTP/2 använder en mera avancerad metod, HPACK, som tar bort överflödig information som finns i HTTP-rubrikpaket och gör inladdningen av en webbsida snabbare. Från varje HTTP-paket tar HPACK bort några byte, som sedan adderas upp till betydliga mängder med tanke på mängden av HTTP-paket som krävs för varje webbsidas inladdning. [2]

REST (eng: *representational state transfer*) har blivit populärt sedan år 2010. API:n (eng: *application programming interface*) kunde nu nå specifika URI:n (eng: *uniform resource identifier*) med HTTP/1.1-metoder istället för att använda nya HTTP-metoder.

Varje webbapplikation kunde utrusta sig med en API för att hämta och modifiera data utan att ändra webbläsaren eller servrarna. Antalet API:n för webbsidor har ökat och tillämpat tillägg sedan 2005. [1]

2.4 Universal Resource Locator, URL

Webbsidor har unika adresser som kallas för URL (eng: *Uniform Resource Locator*). URL:en används för att nå hemsidan. På samma sätt som en fysisk adress eller ett telefonnummer är bundet till en plats eller en person så är en URL bunden till en webbsida. [15]

URL utvecklades från UDI (eng: *Universal Document Identifier*). Tim Berners-Lee presenterade UDI till IETF (eng: *Internet Engineering Task Force*). IETF ändrade *Universal* till *Uniform*, eftersom tekniken inte egentligen var universell. *Document* ändrades till *Resource* på grund av att *Document* var för specifikt. Istället för *Identifier* ville IETF att det skulle vara *Locator* för att få den att låta mindre pålitlig. I grund och botten ville IETF att folk hellre skulle använda URN (eng: *Uniform Resource Name*) istället för URL till 'namn' för att identifiera någonting. [29]

2.5 JavaScript, JS

JavaScript är ett skriptspråk som fokuserar på objekt och deras egenskaper, och används för att skapa dynamiska och interaktiva webbsidor eller applikationer. Det kallas ofta för *the Language of the Web*, eftersom det används på nästan alla webbsidor idag. Bland alla programmeringsspråk som finns idag har JavaScript den mest aktiva gemenskapen med nästan dagliga uppdateringar eller nya funktioner, och mängder med diskussioner i nätforum.

Utvecklare på Netscape Communicator märkte ett problem i webbsidorna under tidigt 1990-tal: de var för statiska. Snart efter att de insett problemet påbörjades utvecklingen av ett skriptspråk som skulle tillåta animationer, formgivning och dynamisk interaktion. JavaScript grundades under namnet Mocha, som byttes till LiveScript år 1995 när den började höra till Netscape Navigator 2.0. Det skrevs och publicerades av Brendan Eich. Namnet byttes senare till JavaScript för att kunna lägga det i samarbete med programmeringsspråket Java, och tack vare den kopplingen fick JavaScript ett fäste samtidigt som Java blev mera och mera populärt. Vartefter att Netscape Navigator blev mera populärt tack vare JavScript började andra webbläsare implementera egna versioner av skriptspråket för att kunna tävla med Netscape. Därmed skrev Microsoft sin egen version av JavaScript, JScript, som började användas till deras webbläsare Microsoft Internet Explorer.

Google publicerade sin webbläsare med öppen källkod (eng: *open-source engine*) Chrome V8 år 2008. Tack vare att den var en högpresterande JavaScript-motor kunde utvecklarna nu bygga applikationer baserade på webbläsare som fungerade lika bra som program på dator och mobil. Samtidigt som Chrome V8 kom ut publicerade Ryan Dahl sin programmeringsram (eng: *environment*) med öppen källkod, Node.js. Den möjliggjorde att man kunde köra JavaScript-kod utanför en browser, vilket ökade mängden möjligheter med JavaScript och har bidragit till JavaScripts stora popularitet idag. Det har totalt kommit 9 versioner av JavaScript under namnet *ECMAScript* (ES). *ECMA International* är ett företag som standardiserar informations- och kommunikationssystem. 1997 tog företaget Netscapes JavaScript och Microsofts JScript och publicerade ECMAScript som standard med specifikationer som båda språken är baserade på. [5][17] [21]

2.6 Ramverk i frontend-programmering

Ramverk (eng: *framework*) och webb bibliotek (eng: *web libraries*) utvecklas till följd av att utseendet och beteendet av webbsidor förändras så snabbt att W3C, som utvecklar och distribuerar nya versioner av HTML, CSS och JavaScript, inte hinner lägga till alla nya funktioner tillräckligt snabbt. Ett ramverk bidrar med en grund på vilken man kan bygga upp en mjukvara utan att behöva börja från noll samtidigt som den hjälper att spara tid och minskar risk för fel. Det finns ramverk för många olika programmeringsspråk. [27]

AngularJS, React och Vue är JavaScript ramverk som används i frontendprogrammering. De tre ramverken finns till för att utveckla användartillgänglighet (eng: *user accessibility*) och skapa ett interaktivt användargränssnitt (eng: *user interface*). En annan sak som de har gemensamt är att de används för att skapa SPA (eng: *single page application*), vilket innebär att den dynamiskt skriver om delar av webbsidan med nya data istället för att ladda om hela sidan. Den här metoden minskar mängden information som servern måste ladda, till exempel vid varje klick, och gör processen snabbare. Tidigare har SPA varit svårt att implementera på grund av att den består av mycket kod, men tack vare CMS (ens: *content management system*) har det blivit lättare idag. CMS är ett program som används för att hantera innehåll, samtidigt som det kan publicera det. Den här funktionaliteten möjliggör att flera användare kan skapa, redigera och publicera innehåll utan en utvecklare. [27][11][23]

React publicerades för första gången år 2013. Det är ett bibliotek med öppen källkod (eng: *open-source library*) som ägs av Facebook och utvecklas hela tiden av företagets utvecklare, samt ramverkets intressegrupp (eng: *community*) bestående av utvecklare och företag. Ramverket röstades till det mest använda ramverket år 2020[7]. Dess främsta funktionsprincip är att komponenter skrivna av olika utvecklare ska fungera bra tillsam-

mans, och att onödiga egenskaper inte läggs till för att undvika onödig kod. [9]

AngularJS var det näst mest populära webbramverket år 2020 [7]. Det publicerades år 2010 som ett sidoprojekt av en Google anställd, men skrevs helt om år 2014. Det är implementerat så att alla komponenter och symboler har en enda uppgift (eng: *single responsibility principle*). [9]

Vue.js eller Vue är ett bibliotek med öppen källkod (eng: *open-source library*) som är anpassningsbart. Det publicerades för första gången år 2013 och idag upprätthålls det av upphovsmannen och en grupp av utvecklare. Vue var år 2020 det tredje mest använda ramverket [7][9]

Django är ett webbutvecklingsramverk med öppen källkod skrivet i Python. Det bidrar med snabb utveckling, och stilren design, och det är säkert, snabbt och skalbart. Rails är ett ramverk som har använts till att utveckla de kända webbsidorna Twitch och Hulu. Ramverket är skrivet i programmeringsspråket Ruby, som har som fokus att förenkla kodning och tack vare det är Rails implementerat att innehålla mindre kod och repetition. [27]

Kapitel 3

Webbsidans design

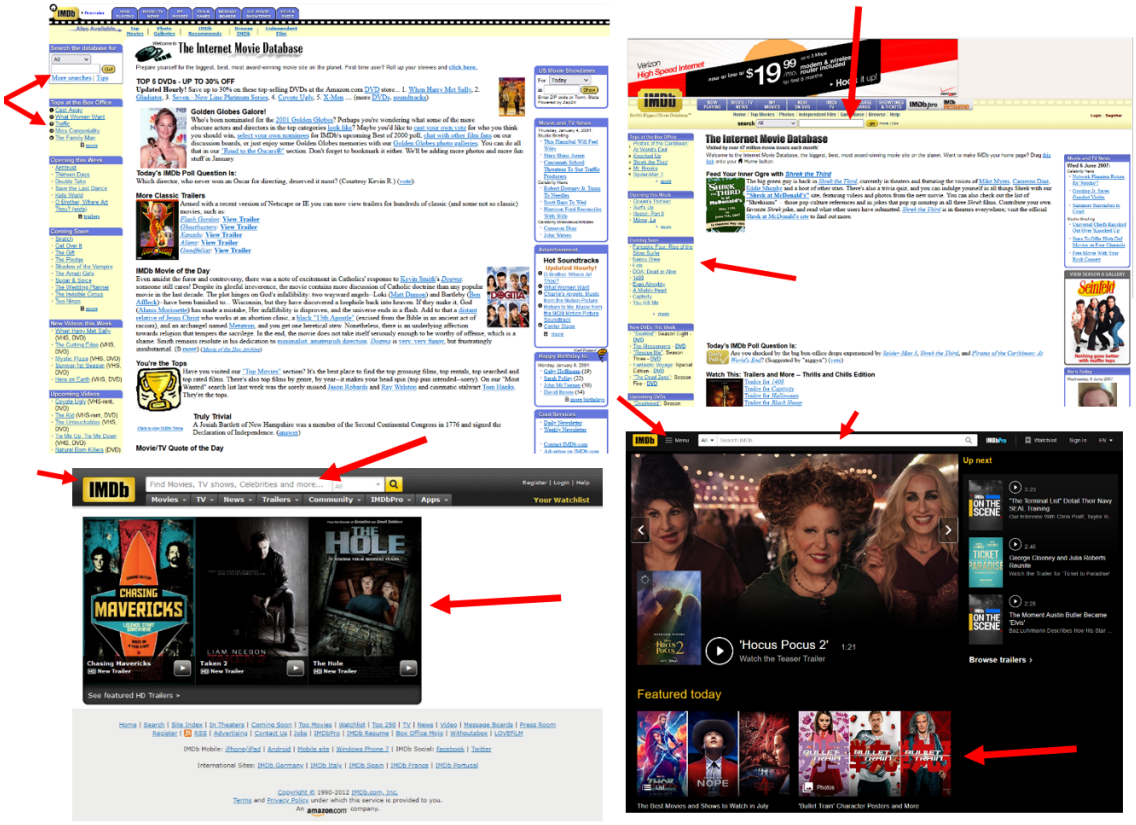
Den första webbsidan bestod av HTML, HTTP och URL och publicerades år 1991. Sedan dess har nya programmeringsspråk publicerats och standardiserats. Idag består webbsidor av förutom HTML, HTTP och URL även av CSS och Javascript. HTML bygger strukturen, CSS bestämmer utseendet och JavaScript justerar webbsidans beteende på användarens sida. Tack vare CSS och JavaScript har webbsidan utvecklats och kraven på webbsidans funktionalitet och design har ändrats flera gånger sedan år 2000.

3.1 En webbsidas design sedan år 2000

Förändringar av designprinciper i webbsidor är inte indelat i perioder på samma sätt som konst kan indelas i epoker, till exempel Barocken eller Gotiken, utan webbdesign sporras av utvecklingen av teknologi, sociala trender och företags målsättningar. Webbsidor i början av 2000-talet innehöll mycket text, vilket kan argumenteras att berodde på att de under en tid var skrivna i endast HTML. Resultatet från en undersökning av Wen Chen et al.[12] år 2016 angående utvecklingen inom webbdesign inkluderar tre specifika markörer som talar för vilken tid en webbsida är från: upplägget av informationsarkitektur, visuella attribut och komposition av media.

3.1.1 Upplägget i webbsidor

Upplägget för informationsarkitektur i webbsidor inkluderar navigationsfältet och andra element som bidrar till att visa användaren hur man navigerar webbsidan och får tag på information. Undersökningen av Wen Chen et al.[12] använde IMDB:s hemsida i sin undersökning, så i den här avhandlingen har motsvarande bilder på webbsidorna sökts upp från webbsidan Internet Archive WayBack Machine och representeras i figur 3.1. I bilderna som representerar IMDB:s webbsida från år 2001 och 2007 visas ett uppmärk-

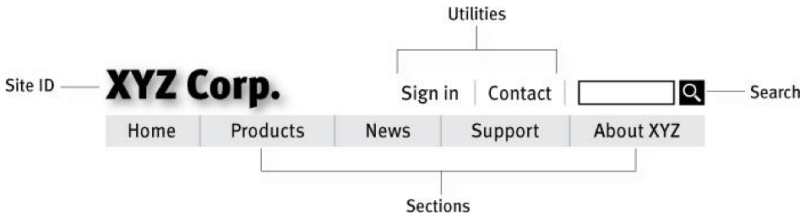


Figur 3.1: IMDB hemsida från år 2001 (övre vänster), 2007 (övre höger), 2012 (nedre vänster) och 2023 (övre höger). Skärmklippen tagna från https://archive.org/web/ .

samhetkrävande navigationsfält med gul bakgrund och blåa länkar som i sin helhet tar upp stora delar av webbsidans yta. I webbsidan från år 2012 har navigationsfältet på vänster sida försvunnit och smugit sig in under sökfältet, och i webbsidan från år 2023 har navigationsfältet gömts undan i ikonen mellan IMDB-logon och sökfältet. I undersökningen noterades det att sökfältet flyttades till ett mera prominent position och blev större efter år 2007, vilket några deltagare påstod att var på grund av att användare har gått från att navigera webbsidan för att hitta information till att söka saker via sökfältet.

Bilder och ikoner har blivit en viktig del av en webbsidas startsida. Logon för en webbsida, som oftast hittas uppe i vänstra hörnet eller uppe i mitten, kallas för en markerande komponent och dess uppgift är att ge användaren en uppfattning om tonen och huvudsyftet för en specifik produkt eller själva företaget. En markerande komponent kan bestå av en eller flera bilder tillsammans med namnet på företaget eller produkten, samt en kort bildtext. Idag använder nästan alla företag en ikon eller bild på sin startsida där användaren lätt hittar den och kan koppla den till produkter eller en service som erbjuds. Figur 3.1 visar att logon var som störst i versionen från år 2012, och att bilderna på webbsidan från år 2012 och 2023 är betydligt större än bilderna från de tidigare åren. [12]

Steve Krug nämner i sin bok *Don't Make Me Think, Revisited* [22] att många saker idag ska hända snabbt och enkelt, som till exempel att hitta saker. Krug talar om att man till exempel i en matbutik förväntar sig att skyltar på sidorna av hyllorna berättar vad som finns i hyllan, eller att det i början av en tidning finns en innehållsförteckning och sidonummer någonstans i kanten av varje sida. Meningen med dessa standarder är att underlätta användningen av och navigering genom sakerna i fråga, till exempel hur man kan navigera sig fram till peston i matbutiken eller lätt få en uppfattning om en tidnings innehåll. När någon av dessa standarder bryts blir man tvungen att tänka om, vilket kan bli frustrerande. Idag ligger ett stort fokus i webbsidor på dess intuitivitet och snabbhet. Till exempel hur snabbt en webbsida tar emot data eller hur snabbt en webbsida laddas in (se stycke 2.3 om HTTP) påverkar en webbsidas intryck på användaren. Krug nämner termen *persistent navigation*, vilket översätts till envis navigation, som står för en samling element som finns på samma ställe på alla sidor i en webbplats. Det används för att ge webbsidan ett bekant och förutsägbart intryck. Krug påstår att den här typen av navigation är bra eftersom den påminner användaren om att man befinner sig på samma webbsida då navigationen finns på samma ställe och ser likadan ut oavsett vilken sida man befinner sig på. Figur 3.2 är tagen från *Don't Make Me Think, Revisited* och representerar de delar av



Figur 3.2: Fyra delar som hör till envis navigation från *Don't Make Me Think, Revisited* på sidan 66 av Steve Krug.

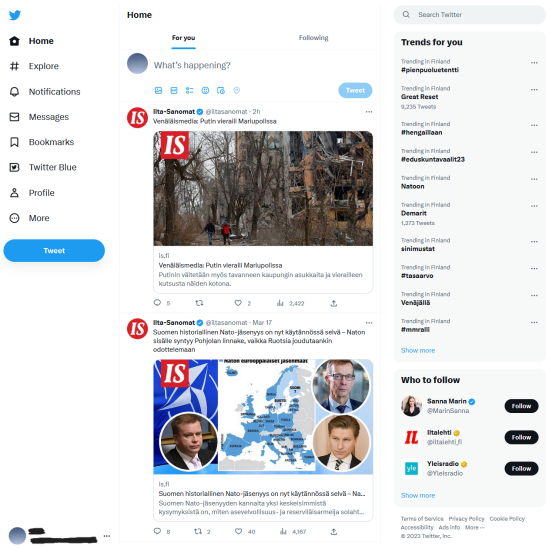
navigationen som hör till envis navigation.

Det finns många olika designsätt att bygga upp sitt navigationsfält som många följer, men det finns inga definitiva regler på hur en webbsidas navigationsfält måste se ut. Det mest kända navigationsfältet idag är horisontellt och räknar upp alla större delsidor av en webbsida. Den här typen av navigationsfält ger användaren en klar uppfattning om vad som finns tillgängligt och var man hittar saker man söker efter med tydliga rubriker. En annan version av navigationsfält är rullgardingmenyer (eng: *dropdown-menyer*). De ser likadana ut som ett horisontellt navigationsfält, bortsett från att det finns rubriker med bredare ämnen som avslöjar underkategorier då man svävar med muspekaren över en rubrik. Rullgardingmenyer används mycket av företag som har många produkter och produktgrupper. Hamburgarnavigationsfält (eng: *Hamburger Navigation Menu*) är en annan typ av design och mest populär i mobilvy eftersom den är en liten ikon som utvidgas när man klickar på den, men göms undan resten av tiden så den ger rum för innehåll under

tiden som man inte navigerar någonstans. Den kallas för *Hamburger Navigation Menu* för att knappen som öppnar menyn ser ut som en liten hamburgare, oftast uppe i någondera av hörnen på mobil- eller datorskärmen. IMDB använder idag Hamburgarnavigation i sina webbsidor både för mobil- och datorskärm (se figur 3.1 på IMDB:s webbsida från år 2023). En vertikal sidonavigation är inte lika populär som en horisontell, men fungerar bra om man har rubriker som är längre och kan vara bra om man har flera huvudrubriker med underkategorier. Den är upplagd så att rubrikerna staplas på varandra på sidan av webbsidan. I vissa webbsidor med horisontellt navigationsfält finns även *footer navigation*, vilket är en extra meny som finns längst nere på webbsidan och där användaren kan hitta saker som inte finns i det horisontella navigationstältet högst uppe på sidan. [16]

3.1.2 En webbsidas visuella attribut

I artikeln av Wen Chen et al. [12] från år 2016 nämns färgskalor, upplägg och kvaliteten av visuella element som bidragande faktorer till hur gammal en webbsida uppfattas att vara. Deltagarna i undersökningen nämner att en webbsida som innehåller för lite färg och mera vita mellanrum uppfattas som platt och minimalistisk. Det är idag en vanlig designstil och används både i ljust och mörkt tema av till exempel Twitter. Figur 3.3 är ett exempel på hur Twitterflödet ser ut i ljust tema. I figuren ser man att det finns mycket mellanrum och endast sökfältet, *Trends for you* och *Who to follow* har en annan bakgrundsfärg. Utöver de tre elementen ser man Twitters logo högst uppe till vänster och under den finns navigationsfältet med ikonen för att lägga till en ny tweet i samma färg som Twitters logo.

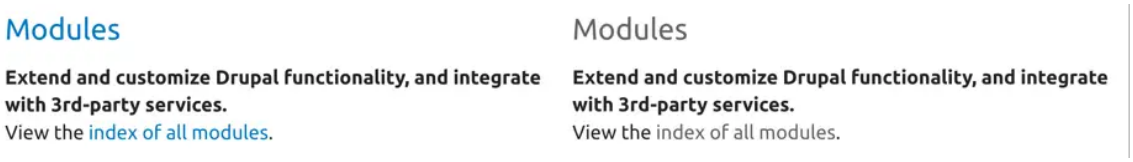


Figur 3.3: Exempel på Twitters webbsida.

Under början av 2000-talet användes många enfärgade rutor från en färgskala. Denna färgskala är ett lätt sätt att identifiera olika perioder för en webbsida. På senare år har färganvändningen blivit mera sparsam och valet av färger ägnas mera eftertanke. De färgade områdena fångar ens uppmärksamhet före de vita områden, speciellt om det finns få färgade områden. Detta gör att företag använder färger till sin fördel och på färre ställen: ju mindre färg de använder, desto lättare är det för kunden att märka de få ställena som faktiskt har färg. Upplägget (eng: *layout*) följer vanligtvis horisontell eller vertikal orientering, eller en blandning av dem. Upplägg är grundläggande när det kommer till hur

intuitiv navigeringen är och hur estetiskt tilltalande webbsidans design är. I undersökningen noterades det att kvaliteten på visuella element på en webbsida hade utvecklats sedan tidigt 2000-tal. Utvecklingen hade inte varit lika påtaglig som upplägget eller färgkompositionerna, men några noterade att kvaliteten i bilder, videor och text hade förbättrats till en mera elegant design eller med högre upplösning. [12]

En webbsidas färganvändning kan vara avgörande till om en användare väljer att stanna på webbsidan, samtidigt som den hjälper att ge användaren rätt intryck angående företagets, eller produktens, ton eller huvudsyfte. I samband med färganvändning måste webbsidor idag satsa på tillgänglighet, eftersom det är en stor faktor i webbsidans popularitet, framgångar och användbarhet eftersom ökad tillgänglighet innebär flera möjliga kunder. Webbsidor kan inte grunda sin läsbarhet på färger eftersom det gör det svårare för färgblinda att använda webbsidan. Länkar i en webbsida kan bli osynliga om en användare med akromatopsi, vilket innebär att användaren inte kan se färg, använder sidan. Det är en designprincip för webbsidor som vill vara tillgännglig för färgblinda att sträcka under länkar, lägga ramar runt knappar som går att klicka på, använda färger med olika konstrast snarare än olika nyans, och använda ikoner för att underlätta navigation. Det finns knep och programvaror som webbutvecklare och webbdesigners använder för att se hur webbsidan ser ut för människor med olika typer av färgblindhet. Ett vanligt trick som vem som helst kan använda är att ta en skärmbild av sin webbsida och ta bort all färg och se om man fortfarande kan urskilja på länkar, knappar och viktiga delar av webbsidan. Med hjälp av olika program kan man generera bilder som visar hur webbsidan ser ut för någon som har svårt att se enskilda färger, som röd, grön och blå. Figur 3.4 visar hur en webbsida ser ut för en människa som ser alla färger till vänster, respektive en människa som inte ser blått till höger. Bilden till höger ser man att inte har någon indikation på att någonting skulle gå att klicka på, trots att *Modules* och *index of all modules* är länkar. Figuren är ett bra exempel på hur länkar inte ska designas: endast markerade med en specifik färg. [14]



Figur 3.4: Bilder som representerar webbsida som den uppfattas för en användare utan färgblindhet och en användare som inte ser blåa färger. Bilderna tagna från https://axesslab.com/colorblind-accessibility-web-fail-success-cases/ .

3.1.3 Komposition av media i en webbsida

I artikeln av Wen Chen et al. [12] från år 2016 påpekade alla experter att det användes betydligt mindre text i en webbsidas startsida idag. Några utvecklare påstår att för mycket text på startsidan kan överväldiga och tråka ut användare. Vidare noterade deltagare att det i början av 2000-talet användes hyperlänkar uteslutande för att signalera att det var någonting att klicka på. Dessa länkar ledde till en annan del av webbsidan eller till en helt annan webbsida. Länkens riktning signalerades endast med text, vilket idag har ersatts med delvis text och delvis ikoner, eller båda i ett. Många deltagare i undersökningen anmärkte att mängden information i en webbsida hade ökat mot 2010-talet och därmed fört med sig krav på hur media skulle framföras. En webbsida som innehåller mycket information lägger mycket vikt i att ha ett intuitivt och välstrukturerat upplägg för att en användare ska förstå hur man hittar specifika saker i webbsidan utan att tappa bort sig i webbsidans struktur. Ökningen i information sporrade i början av 2000-talet ett större fokus på sökfältet i webbsidor (se stycke 3.2.1) för att underlätta navigeringen via en lång sökväg (eng: *path*) till det man söker efter från startsidan. Dock drogs slutsatsen att en ökning i information inte är en ensam faktor till att sökfältet har blivit så mycket mera använt sedan tidigt 2000-tal eftersom sökfältet även var en prominent del av mindre webbsidor med mindre information. Sökfunktionernas utveckling och användbarhet bidrog till att flera och flera webbsidor använde dem. Utvecklingen av hårdvara understöddes av snabbare nätverk och möjliggjorde överföring av större filer och bilder, som kunde visas på olika skärmar tack vare skärmutvecklingen. Efter att mobiltelefonen blev populär ökade kraven på webbsidorna, eftersom de nu också skulle kunna visas och läsas på en liten skärm. Med hjälp av anpassningsbar design (eng: *adaptive design*) kan skärmar idag justeras i storlek och upplösning, samt anpassa position av element beroende på vilken skärm man har. [12]

Kapitel 4

Frontend-utveckling

4.1 Frontend-utveckling av webbsidor idag

På grund av webbens storlek och mängder av webbsidor som finns tillgängliga på den är det viktigt för varje webbsida att sticka ut för att få trafik till den. Några företag eller privatpersoner försöker nå människor via sociala medier, till exempel genom att länka till sin webbsida i sin Instagram- eller Twitter-profil, och vissa gör det genom att publicera reklam på andra webbsidor som många besöker. Det finns även företag som samarbetar med andra företag för att få publicitet och förhoppningsvis locka någon som är intresserad av samarbetspartnern att även bli intresserad av ens eget företag.

Bortsett från att locka användare till ens webbsida via sociala medier eller reklam kan man utnyttja sökalgoritmer för att öka sannolikheten att människor ser och når ens webbsida. Ett mycket använt koncept idag är SEO (eng: *search engine optimization*). SEO är ett samlingskoncept för strategier man använder för att få sin webbsida att placeras högre upp i sökmotorer, till exempel Google. Genom att analysera hur sökalgoritmen tolkar en webbsida kan man manipulera och designa sin webbsida enligt kraven. Det finns tre kärnmått som sökalgoritmen använder för att evaluera en sidas kvalitet: först analyseras antalet länkar från andra webbsidor till webbsidan i fråga, sedan söker algoritmen genom innehållet av webbsidan för att kunna anpassa sökord till den, och till sist analyseras sidstrukturen i webbsidan, vilket inkluderar rubrikerna och URL:er i webbsidan. [3]

En artikel skriven av Ben Hofferber [19] handlar om hur programmering av webbsidor är annorlunda idag jämfört med tidigt 2000-tal. Han tar fram att det används massvis med olika ramverk från olika programmeringsspråk idag, och bland dem är de mest använda ramverken skrivna i JavaScript eller Python. Enligt en artikel av Atlantic Press år 2016 [4] ska utvecklare idag ha, förutom kunskap i HTML, CSS och JavaScript, även ha höga kunskaper inom optimering av nätverksegenskaper (eng: *network property optimization*), förstå sig på sökotorers klassificeringsstöd, veta hur man använder verktyg som hör till

stödutveckling (eng: *auxiliary development*) och ha erfarenhet med kodupprätthållning. Dessa två tyder på växande krav i fördjupning samt bredd av kunskaper när man ska arbeta med webbutveckling. Frontend-programmering kan tyckas idag ha för många alternativa verktyg och ramverk att välja mellan.

I en blogg av Nick Janetakis från år 2018 diskuteras förändringen av webbutveckling sedan 2000-talets början från en webbutvecklare som har jobbat med resurserna från sent 90-tal såväl som resurserna som finns tillgängliga idag. Janetakis nämner att det finns mycket information på nätet som hjälper programmerare och utvecklare att hitta lösningar, samt lära sig av andras misstag. Han anser att dagens minskning i produktivitet är en konsekvens av detta överflöd av information och att man lätt kan fastna i en cirkel av att försöka hitta en perfekt lösning istället för att testa sig fram och få någonting gjort. Webbutveckling var till en början inte indelat i frontend- och backend-programmering på samma sätt som det är idag, utan då var en utvecklare som en person som utvecklar och designar webbsidor oberoende vilket programmeringsspråk hen använder. Designförändringar av en webbsida var inte lika lätt som det är idag. En webbsida design byggdes oftast upp i ett program som Photoshop och när designen verkade okej så klippte man upp den i mindre delar som sedan sammanfogades med hjälp av HTML till en webbsida. Om man ville ändra färgen på webbsidan, eller en specifik bild, eller till exempel runda av hörnen på bilderna, var man tvungen att editera helheten i photoshop, klippa upp den i mindre bilder, ladda ner de nya bilderna och sen ladda upp de igen med hjälp av HTML. Eftersom CSS inte stöddes under den tiden var utvecklare tvugna att lägga till skärmbredd i form av extra pixlar vid sidorna för bredare skärmar, och detta gjordes genom att lägga ett flertal pixel-breda bilder bredvid varandra. Janetakis nämner att relativt stora förändringar, som att ta bort ett navigationsfält eller att helt bygga upp enn ny webbsida, inte tar lika länge eller lika mycket jobb idag som det gjorde under tidigt 2000-tal. Responsiv design var inte möjligt under tiden före JavaScript och välanpassad CSS, utan webbsidorna var väldigt statiska. [20]

Idag handlar webbutveckling mycket om förarbetet, vilket enligt Janetakis består av val av ramverk, bibliotek, tillägg, underlydande områden (eng. *dependencies*), mallspråk (eng: *templating language*), lära sig hur man lägger upp en rutt för URL, lära sig hur man konfigurerar en webbserver, etc. I en artikel av Lisa Plitnichenko angående webbutveckling de 20 senaste åren tas webbutvecklingen fram som enkel idag och endast bestående av tre steg: koda och spara din kod, öppna filerna lokalt i en webbläsare för att se hur de ser ut, och till sist publicera webbsidan genom att lägga tilferna i ett program som Github Pages eller Netlify. Hon nämner också att med utvecklingen av webben har flera krav dykt upp och blivit mera komplexa, och tack vare utvecklingen av JavaScript och dess ramverk har webbutveckling såväl blivit mera effektivt som mera komplext. Dessa

motsägande artiklar, Janetakis påståenden att webbutveckling är för komplicerat och att det finns för mycket information, och Plitnichenkos anspråk på mängden med nya möjligheter och funktionaliteter i webbsidor, har båda bra argument. Janetakis har rätt i att webbutveckling är mera komlicerat på grund av mängden information. Det har blivit svårare att veta vilket ramverk eller bibliotek man ska välja, eftersom det finns så mycket, och man måste generellt sett ha mycket bredare kunskap samtidigt som man måste ha djupa kunskaper. Plitnichenko har rätt i att webbutveckling är lätt med tanke på att det faktiskt finns verktyg till för nästan alla problem och funktionaliteter man har i sin webbsida. Man kan inte på ett lätt sätt säga vem som har mera rätt när det kommer till dagens överflöd av information. Man kan tänka sig att en relativt ny programmerare, som endast har kodat i HTML, CSS och JavaScript, inte kan tala för hur det var att koda förut, menhoppet finns att dagens mängder med information för med sig så mycket spännande möjligheter för personalisering, skräddarsydda användargränssnitt och möjlighet att göra nästan precis var man vill med en webbsida. Samtidigt har det blivit lättare att fixa en webbsida med färdiga mallar och verktyg och på det sättet spara tid. [26]

4.2 Webbsidan i framtiden

Sedan sent 90-tal har webbsidor utvecklats, både i utseende och funktionalitet, vilket innebär att även programmeringen har utvecklats. Från att utveckla webbsidor under långa tidsperioder med HTML tillsammans med en version av CSS som inte ännu var anpassat till alla webbläsare, till att idag kunna bygga upp en webbsida med hjälp av färdigbyggda (eng; *pre-built*) komponenter och verktyg på en bråkdel av tiden och med betydligt säkrare och mera skräddarsydda resultat.

Ramverk och bibliotek med öppen källkod (eng: *open-source libraries*) möjliggör idag snabbare utveckling och bättre anpassning till både användarens och utvecklarens behov. På grund av att den standardiserade utvecklingen av HTMl, CSS och JavaScript sker av W3Cs (eng:*World Wide Webb Consortium*), och till följd av att besluten kräver detaljerade processer, sker de grundläggande språkens förändringarna mycket långsamt i jämförelse med utvecklingen av krav i webbsidors design och funktionalitet. Eftersom flera utvecklare med olika specialiseringar och kunskaper kan jobba tillsammans och analysera vad som saknas i ett programmeringsspråk, vilka brister det finns i existerande komponenter och optimera särskilda verktyg eller element, sker utvecklingen så mycket snabbare i den ostandardiserade webbutvecklingen.

Det spekuleras det om utvecklignen och uppdateringar av gamla funktioner kommer att växa lika snabbt om några år, eller om utvecklingen kommer att sakta ner när teknologin når ett tak där ingenting längre kan uppfinnas eller endast minimala optimeringar är

möjliga. Orsaken till att det finns människor som är tveksamma mot att det skulle finnas en gräns för teknologins utveckling är att det hittills har skett så drastiska förändringar att det är svårt att se var gränsen för möjliga teknologier går. Webbsidans utveckling har tagit massiva utvecklingssteg sedan 90-talet och utvecklingen för JavaScript idag ser relativt lovande ut på grund av dess aktiva intressegrupper. Det framstår som att JavaScrip publicerar nya bibliotek, verktyg och ramverk nästan dagligen, men trots dess populära position finns det alltid en chans att det i framtiden ersätts av ett snabbare skriptspråk med flera funktioner, på samma sätt som programmeringsspråk från 1980-talet ersatts av nyare programmeringsspråk idag. Motargumentet till att JavaScript, HTML eller CSS, skulle försvinna är att de är de mest använda frontend-programmeringsspråken idag och att alla webbsidor är grundade på ett HTML-dokument. Det finns dock genvägar som ger möjligheten att utveckla webbsidor utan att behöva skriva någon kod i HTML eller CSS. Funktioner och mallar (eng: *templates*) som har element som motsvarar HTML- och CSS-kodsnuttar kan användas istället. Man kan idag konstatera att alla webbsidor är beroende på HTML och CSS, samtidigt som nästan alla webbsidor använder någon form av JavaScript-skript för att öka användarvändligheten och förbättra användargränssnittet (eng: *user interface*). Slutligen så är det ingen som kan förutspå framtidens utkomst, utan det enda man kan göra är att kolla tillbaka på historien och skapa hypoteser om framtiden. Men det bästa man kan göra är att fokusera på nuet, vilket innebär webbutveckling med ett brett utbud av ramverk, bibliotek och programmeringsspråk, samt diverse principer för tillgänglighet och användarvänlighet i design. [5]

Kapitel 5

Avslutning

[TODO: To be continued....]

Litteraturförteckning

[1] Evolution of http. URL https://udn.realityripple.com/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP. Läst 26.3.2023.

[2] What is http? why is http/2 faster than http/1.1? URL https://www.cloudflare.com/learning/performance/http2-vs-http1.1/. Läst 26.3.2023.

[3] What is search engine optimization. URL https://www.optimizely.com/optimization-glossary/search-engine-optimization/. Läst 11.3.2023.

[4] Exploration of web front-end development technology and optimization direction, 2016. URL https://www.atlantis-press.com/proceedings/icence-16/25860891. Läst 11.3.2023.

[5] A brief javascript history, 2023. URL https://launchschool.com/books/javascript/read/introduction. Läst 9.3.2023.

[6] What is css?, 2023. URL https://www.w3.org/Style/CSS/Overview.en.html. Läst 21.2.2023.

[7] State of js 2020, 23.9.2021. URL https://web.archive.org/web/20210411021418if_/https://2020.stateofjs.com/en-US/technologies/front-end-frameworks/. Läst 9.3.2023.

[8] An overview of http, 3.3.2023. URL https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview. Läst 9.3.2023.

[9] Mitra Bauer. Hello framework! a heuristic method for choosing front-end javascript frameworks, 2021. URL https://uu.diva-portal.org/smash/get/diva2:1591850/FULLTEXT01.pdf. Läst 9.3.2023.

[10] Tim Berners-Lee. Hypertext markup language, 1995. URL https://www.rfc-editor.org/rfc/rfc1866. Läst 20.2.2023.

[11] Donna-Marie Bohan. What is a content management system and how does it work?, 1.2.2023. URL https://www.bloomreach.com/en/blog/2019/content-management-system. Läst 9.3.2023.

[12] W. Chen, D. Crandall, and N. Makoto Su. Understanding the aesthetic evolution of websites: Towards a notion of design periods, 2017. URL https://vision.soic.indiana.edu/papers/webevolution2017chi.pdf. Läst 11.3.2023.

[13] cloudflare.com. What is http?, 2023. URL https://www.cloudflare.com/learning/ddos/glossary/hypertext-transfer-protocol-http/. Läst 26.2.2023.

[14] DesignMantic. Web design guidelines for color blind audience, 2023. URL https://www.designmantic.com/community/website-design-guide-color-blind.php. Läst 24.3.2023.

[15] encyclopedia.com. Three protocols, 2023. URL https://www.encyclopedia.com/economics/encyclopedias-almanacs-transcripts-and-maps/three-protocols. Läst 30.2.2023.

[16] Anna Fitzgerald. Website navigation: The ultimate guide [types top examples], 27.2.2023. URL https://blog.hubspot.com/website/main-website-navigation-ht. Läst 12.3.2023.

[17] Kathryn Frid. Essay on history of javascript, 2021. URL https://bldl.ii.uib.no/2021/INF328B_HOPL4_essays.pdf#page=37. Läst 9.3.2023.

[18] Jasmine Harwood. The evolution of html, 2018. URL https://medium.com/@jasmineharwood/the-evolution-of-html-837f85e6c1ee. Läst 20.2.2023.

[19] Ben Hofferber. A brief history of web programming, 2022. URL https://rangle.io/blog/a-brief-history-of-web-programming. Läst 9.3.2023.

[20] Nick Janetakis. Was web development better back in the early 2000s?, 13.3.2018. URL https://nickjanetakis.com/blog/was-web-development-better-back-in-the-early-2000s. Läst 28.3.2023.

[21] Christina Kopecky. Javascript versions: How javascript has changed over the years, 18.12.2020. URL https://www.educative.io/blog/javascript-versions-history. Läst 27.3.2023.

[22] Steve Krug. *Don't Make Me Think, revisited (Third Edition)*. New Riders, 2014. ISBN 978-0-321-96551-6.

[23] Katie Lawson. What are single page applications and why do people like them so much?, 16.9.2022. URL https://www.bloomreach.com/en/blog/2018/what-is-a-single-page-application. Läst 9.3.2023.

[24] Frank Moraes. Html for beginners the easy way: Start learning html css today, 2023. URL https://html.com/#The_History_of_HTML. Läst 27.2.2023.

[25] Priya Pedamkar. Versions of html, 2023. URL https://www.educba.com/versions-of-html/. Läst 25.3.2023.

[26] Lisa Plitnichenko. How web development has changed in the last 20 years, 2020. URL https://jellyfish.tech/blog/web-development-evolution-from-2000s-to-2020/. Läst 28.3.2023.

[27] Codeacademy Team. What is a framework?, 23.9.2021. URL https://www.codecademy.com/resources/blog/what-is-a-framework/. Läst 9.3.2023.

[28] Kushleen Waraich. What are the major difference between css, css2 and css3. URL https://www.codingninjas.com/codestudio/library/what-are-the-major-difference-between-css-css2-and-css3. Läst 25.3.2023.

[29] Tantek Çelik. How url started as udi, 2014. URL https://tantek.com/2014/304/b1/url-started-as-udi-conversation-w3c-tpac. Läst 27.2.2023.