

Webbutveckling 1

med HTML och CSS

Med övningar och
projektuppgifter

Förlag: Lieta AB

Titel: Webbutveckling 1 med HTML och CSS

Författare: Taifun Alishenas
 info@taifun.se

Copyright © 2024 Lieta AB
All rights reserved

Maj 2024



Kopieringsförbud!

Denna bok är skyddad av Lagen om upphovsrätt. Kopiering är förbjuden. Förbudet inkluderar översättning, tryckning, stencilering, kopiering, lagring i elektroniska och digitala media, visning på bildskärm eller via projektor, bandinspelning osv. Dessa förbud gäller även för koden i alla programexempel samt övningarnas lösningar som finns i boken. Den som bryter mot lagen om upphovsrätt kan åtalas av allmän åklagare och dömas till böter eller fängelse i upp till två år samt bli skyldig att erlägga ersättning till upphovsman/rättsinnehavare.

Innehåll

Ämne

Sida

Program

Kapitel 1 **Introduktion till webbutveckling** **6**

1.1 Om webbutveckling	7	
- De tre skikten	8	
- Programmeringstänkande	8	
- Algoritmer	9	
1.2 Webbutvecklingens miljöer	11	
- Editorer / IDE	11	
- Interpretator vs. kompilator	11	
- Webbläsare	12	
- HTML – webbens standardspråk	12	
- Script vs. program	13	
- Filändelser	14	
1.3 Att komma igång med HTML	15	
- Vårt första HTML-script	15	Welcome
- Kommentarer	16	
- HTML-taggar / HTML-element	16	
- En övergripande struktur	17	
- Headers	17	Headers
- Utskrift i flera rader	18	Break
- br -taggen	19	
Frågor till kap 1	20	
Övningar till kap 1	22	

Kapitel 2 **Grundbegrepp i webbutveckling** **23**

2.1 Länkar	24	Links
- Elementet ankare	25	Contact
- Attribut	26	
- Varför heter det ankare?	26	
2.2 Bilder	27	Picture
- – ett tomt element	27	
- Förkortningsregeln för tomma element	28	
- Val av bildstorlek	29	
2.3 Bilder som länkar	30	Nav
- Nästlade element	31	
- Navigeringsmeny	31	
- Interaktion	32	
2.4 Specialtecken	33	Contact2
- Namnkoder för specialtecken	33	

Ämne	Sida	Program
- Talkoder för specialtecken	34	
- <hr> -taggen	35	
2.5 Punktlistor	36	Links2
- Elementet unordered list 	36	
- -taggen	36	
2.6 Nästlade och ordnade listor	38	List
- Elementet ordered list ol	38	
- -attributet type	39	
- Nästlade listor	39	
- God programmeringsstil	39	
Frågor till kap 2	41	
Övningar till kap 2	44	

Kapitel 3	Mer om HTML	46	
3.1 Tabeller		47	Tabell1
- <table> -attributet border		47	
- Elementet table		48	
- Elementen th och td		48	
3.2 En mer utvecklad tabell		49	Tabell2
- th- & td- attributen rowspan & colspan		50	
3.3 HTML Forms		51	Form1
- Elementet form		51	
- JavaScript funktion		52	
- Textboxar med elementet input		52	
3.4 En mer utvecklad Form		53	Form2
- Elementet textarea		54	
- Maskerad textbox		54	
- Checkboxar		54	
3.5 Radioknappar och Dropp-down list		55	Form3
- Radioknappar		55	
- Dropp-down list		57	
3.6 Interna länkar		58	Intern_Links
- Namngivna ankare		58	
3.7 Interna länkar i andra dokument		61	Extern_Link
- Sökväg som referens		61	Intern_Links_2
3.8 Image maps		64	Picture
- Elementet map		64	
- Elementet area		65	
- Koordinatsystem för geometriska figurer		65	
3.9 Framesets		66	Index
- Elementet frameset		66	

Ämne	Sida	Program
- Elementet frame	66	
- Navigeringsmeny i en frame	66	
Övningar till kap 3	68	
Kapitel 4 Cascading Style Sheets (CSS)	71	
4.1 Inline Styles	72	
- De tre skikten	72	
- Vad är CSS?	72	
- Historien	72	
- Scriptet Inline	73	Inline
4.2 Internal Styles	74	
- Embedded Style Sheets	74	
- CSS-elementet style	74	
- Style class och attributet class	74	
- Scriptet Declared	74	Declared
4.3 Conflicting Styles	75	
- Överskuggning (Overriding)	75	
- CSS överskriver HTML	75	
- Pseudoklass	75	
- Scriptet Advanced	75	Advanced
4.4 External Styles	77	
- Elementet link	77	
- Extern CSS Style Sheet	77	
- Scriptet External	77	External
4.5 Absolut positionering	79	Positioning
- Attributet position	79	
4.6 Relativ positionering	80	Positioning2
- Elementet span	80	
Övningar till kap 4	81	

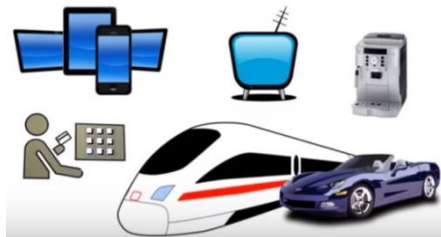
Kapitel 1

Introduktion till Webbutveckling

Ämne	Sida	Program
1.1 Om webbutveckling	7	
- De tre skikten	8	
- Programmeringstänkande	8	
- Algoritmer	9	
1.2 Webbutvecklingens miljöer	11	
- Editorer / IDE	11	
- Interpretator vs. kompilator	11	
- Webbläsare	12	
- HTML – webbens standardspråk	12	
- Script vs. program	13	
- Filändelser	14	
1.3 Att komma igång med HTML	15	
- Vårt första HTML-script	15	Welcome
- Kommentarer	16	
- HTML-taggar / HTML-element	16	
- En övergripande struktur	17	
- Headers	17	Headers
- Utskrift i flera rader	18	Break
- br-taggen	19	
Frågor till kap 1	20	
Övningar till kap 1	22	

1.1 Om webbutveckling

Världen vi lever i är full med prylar som kallas för "intelligenta". Man talar om *artificiell intelligens*. Men prylarna kan inte tänka själva. Någon har programmerat dem, närmare bestämt de elektroniska komponenterna i dem – små datorer som styr all funktionalitet



och interaktivitet. De programmerade små prylarna kallas för *embedded systems*. Är de dessutom uppkopplade mot Internet pratar man om *Internet of Things (IoT)*, resultat av en utveckling som tog fart på 90-talet och sedan dess har revolutionerat alla områden i det sociala livet och människans sätt att leva i hela världen – på gott och ont.

Programmering är ett av de mest spännande kapitlen i teknologihistorien. Inte bara därför att den har lagt grunden till den moderna IT-industrin. Den har bidragit till att förverkliga den urgamla mänskliga drömmen att förenkla mödosamma arbeten. Istället för att plåga sig instruerar man en maskin med idéer och låta den göra jobbet, för att ha mer tid över för annat roligt i livet. Det är roligare att köra en bil än att bara åka med. Det är kreativiteten och det fria skapandet som lockar. Man kan testa helt nya egna idéer.

Webbutveckling är en speciell form av programmering, bara att språket man kodar med – och även resultatet man får, är lite annorlunda. När man tröttnat på att använda program som andra skrivit – surfa, chatta, maila eller lyssna på musik – är det dags att börja programmera själv. Och varför inte skapa och publicera en egen webbsida?

Vad behöver man för webbutveckling?

Mindre än för programmering. Det räcker med en dator samt följande:

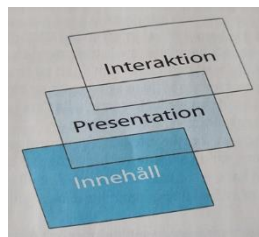
1. Lite programmeringstänkande
2. En vanlig texteditor
3. En webbläsare (web browser)
4. Ett bildbehandlingsprogram (helst)

För 1 läs vidare! För 2-4 spelar det ingen roll om du har en Mac-, Windows- eller Linux-dator. En editor och en webbläsare finns förinstallerade på alla datorer. Annars rekommenderas Google Chrome som webbläsare som kan gratis laddas ned. Andra exempel ges senare (sid 11).

De tre skikten

Man brukar tala om tre olika skikt när det gäller webbsidor. Och det rekommenderas att hålla isär dessa och behandla dem till tre olika webbutvecklingsvertyg:

- *Innehållet* kodas med **HTML**.
- *Presentationen* formges med **CSS**.
- *Interaktionen* programmeras med **JavaScript**.



Verktygen kommer att tas upp i detalj senare. De utgör huvudinnehållet i kurserna webbutveckling 1 och 2.

Förr i tiden blandades dessa tre skikt, vilket ledde till svårigheter. Under tiden har man lärt sig att modularisera, dvs bryta ned och strukturera utvecklingsarbetet genom att separera de här tre skikten. Som kriterium för en webbsidas kvalitet anses primärt sidans användarvänlighet. Kommunikationen med användaren står i centrum. Andra viktiga kriterier är enkelhet och möjligheten att underhålla och uppdatera sidan.

Programmeringstänkande

*"Everyone in this country should learn how to program a computer. Because it teaches you how to **think**."* Steve Jobs

Hur programmerar man?

Egentligen gör vi det varje dag utan att vara medvetna om det. Är t.ex. en lampa trasig hemma följer vi ungefär ett handlingsförlopp som kan beskrivas med bilden till höger, ett s.k. *flödesschema*. I praktiken löser vi problemet att ersätta en trasig lampa genom att tänka och göra så utan att någonsin rita ett flödesschema.

Flödesschemat illustrerar och dokumenterar dock *algoritmen*, dvs tillvägagångssättet för problemets lösning. När den en gång är ritad skulle den kunna användas av vem som helst



som vill byta en trasig lampa. Den blir en slags allmängiltig manual för just detta problem. Men ännu viktigare är att metodiken kan tas över till svårare problem.

Ett annat vardagligt exempel är matlagning. Vare sig vi använder ett recept ur en kokbok eller lagar efter känsla, följer vi en algoritm som dessutom – till skillnad från lampalgoritmen – även har en *input*, råvaror och en *output*, maträtten. Hårdvaran som hjälper oss är köket med alla sina instrument. Matreceptet är mjukvaran dvs programmet. Det är precis samma struktur när vi kör ett program på datorn, matar in indata och får ut utdata som resultat. Programmet vi använder är avgörande för resultatet, precis som matreceptet samt dess förverkligande är avgörande för om vi lyckas med maträtten.

Algoritmer

Båda exemplen visar: Det är algoritmer som medvetet eller omedvetet styr *hur* vi gör – ett sätt att tänka vars gemensamma drag kan generaliseras så här:

1. Att formulera problemet och definiera målet. Hur når vi målet – problemlösningens lösning?
2. Genom att bryta ner problemet i mindre, överskådliga och enklare delar, s.k. *moduler*. Varje modul ska i princip kunna utföras av vem som helst. Detta kallas för *modularisering* som är en allmän princip inte bara i webbutveckling utan i all problemlösning.
3. Genom att ge *instruktioner* som leder till problemets lösning. De måste formuleras på ett entydigt sätt så att de inte kan tolkas på olika sätt. För datorer gör exakt som vi säger. Det har visat sig att det vanliga språket inte lämpar sig för detta ändamål, för det är tolkbart. Skönlitteraturen är ett praktexempel för olika tolkningar av språket. Det vore synd om det inte vore så. Därför har man i webbutveckling hittat på andra, speciella språk vars vokabulär och syntax följer strikta regler som är entydiga. Datorn kan tolka dessa regler endast mekaniskt.
4. I denna process uppstår situationer där vi måste träffa ett *val* – samma sak som att besvara en *fråga*. Den första frågan i algoritmen "Att byta lampa" är "Är lampan inkopplad?" (ovan). Valet mellan "Ja" och "Nej" avgör hur algoritmen fortsätter. Ytterligare val följer.

Det är avgörande att skilja mellan *instruktion* och *val*. En instruktion är ett *kommando* som måste *utföras* medan ett val är en *fråga* som måste *besvaras*. I flödes-

planen till lampalgoritmen är *instruktion* (grön) och *val* (gul) markerade med olika färger. Deras distinktion blir avgörande när man går över från flödesplan till kod.

Algoritmers byggstenar

Man delar in algoritmers viktigaste ingredienser i tre kategorier och kallar dem för *kontrollstrukturer*, eftersom de är generella strukturer som styr och kontrollerar algoritmerna och ger dem den karakteristiska ordningen. Dessa grundläggande kontrollstrukturer är *sekvens*, *selektion* och *repetition* och kommer att tas upp i boken. De anses vara algoritmers byggstenar. Alla algoritmer är uppbyggda av dem.

Avgörande för en algoritms funktionalitet är ingrediensernas *inbördes ordning*. Tar man in i en kokande gryta potatisen först och köttet sedan – istället för tvärtom – blir det mos istället för maträtt. I detta sammanhang hör även algoritmens korrekta avslutning. Utan ett exakt formulerat *avslutningskriterium* som uppnås i ändlig tid uppstår evighetsloopar. När sådant inträffar brukar vi ofta säga att datorn "hängt sig". I själva verket är orsaken en algoritm med ett inkorrekt konstruerat avslutningskriterium. Allt detta kommer att behandlas utförligt i boken.

Ytterligare en ingrediens av algoritmer är *logik*. Datorer kan ingen logik. Människan måste föra över logiken in i datorn. Det är det som kallas för *artificiell intelligens*. Bl.a. formuleringen av korrekta avslutningskriterier i val och loopar, men även modularisering och strukturering kräver logiskt tänkande.

Att upptäcka *mönster* är också en förmåga som ofta behövs i konstruktion av algoritmer, vilket vi kommer att se i våra programexempel som följer i boken.

I valet av instruktioner som ska tas med i en algoritm är det en självklarhet att man sorterar bort allt som är mindre relevant och tar in endast det som är relevant. Dvs även att avgöra *relevansen* av saker och ting för att uppnå det definierade målet (punkt 1) hör till webbutvecklarens uppgifter.

1.2 Webbutvecklingens miljöer

Webbutveckling är i allra högsta grad ett praktiskt ämne.

Man kan inte lära sig webbutveckling genom att endast läsa böcker. För att lära sig webbutveckling måste man skriva kod och testa koden – precis som när man lär sig att köra bil. Och för att göra det behöver man en miljö, där man kan skriva och en annan där man kan köra och testa koden. Den första är en texteditor, den andra en webbläsare (browser). Det finns en uppsjö av utvecklingsmiljöer för de olika språken. Ofta kallas de för IDE, *Integrated Development Environment* – nästan för avancerade för oss. En enklare variant – ofta en del av en IDE – är en *editor*.

Editorer

En *editor* är ett skrivverktyg på datorn, dvs ett program som kan hantera text. *Ordbehandlingsprogram* är en annan beteckning på editorer. På de flesta datorerna finns minst en editor förinstallerad. För att skriva kod och spara den i en fil behövs en editor. Men kod får innehålla endast sådana tecken som kan 'förstås' av webbutvecklingsspråket. Därför måste editorn spara filen som *oformaterad textfil*, dvs utan styr- och kontrollkoder som i vissa ordbehandlingsprogram används för formatering (typsnitt, stil, sorlek osv.). Ett exempel på sådana program är Word som formaterar texten och sparar sina filer som dokument av typ *.docx. Formatering innebär att det läggs till osynliga tecken i texten som webbutvecklingsspråket inte känner till. Motsvarigheten på Mac-datorer är Pages. Sådana ordbehandlingsprogram är **inte** lämpliga för att skriva kod. Däremot kan t.ex. Notepad (Anteckningar), Notepad++ eller TextPad på Windows-datorer och Textredigerare, TextEdit eller Emacs på Mac-datorer vara lämpliga texteditorer för webbutveckling och programmering, eftersom de sparar alla filer som rena textfiler *utan* formatering. För textfiler kan filändelser av typ *.txt väljas, men även andra, beroende på operativsystemet.

IDE står för *Integrated Development Environment*, är alltså en integrerad programutvecklingsmiljö som inkluderar en editor, en *interpretator* resp. *kompilator* och andra verktyg för programutveckling i en och samma samlad miljö. *Visual Studio* är ett exempel på en IDE som har utvecklingsverktyg för ett antal språk. Men JavaScript behöver ingen IDE. Det räcker med en texteditor där koden skrivs och sparas samt en webbläsare där koden exekveras. Vi kommer att använda oss av denna möjlighet som är oberoende av tredje parts verktyg för att slippa installera nya program.

Interpretator vs. kompilator

En *interpretator* är ett program som *tolkar* källkod till maskinkod och skickar maskinkoden till datorns processor utan att mellanlagra den på hårddisken. Processorn exekverar maskinkoden. Källkod är kod som endast människan förstår, men inte

datorn. Maskinkod är kod som endast datorn förstår, men inte människan. Alla webbutvecklingsspråk är interpreterande, inkl. HTML.

Till skillnad från en interpretator är en *kompilator* ett program som *översätter* källkod till maskinkod och lagrar maskinkoden på hårddisken. Först när man exekverar skickas den kompilerade maskinkoden till datorns processor och utförs där. Vissa programmeringsspråk är kompilerande (C/C++), andra är interpreterande (Python).

Webbläsare

En *webbläsare*, på engl. *web browser* är ett program som kan *tolka* HTML-kod samt andra scriptspråkens koder. Att *tolka* är synonym till att *interpretera*, *exekvera* eller *utföra*. Webbläsaren är alltså i huvudsak en HTML-interpretator. Själva programmet har skrivits i något av de universella språken, ofta i C. Exempel på sådana program är Google Chrome, Internet Explorer, Firefox, Safari, Netscape, En webbläsare finns förinstallerad på alla datorer.

HTML-kod kan exekveras av webbläsaren både lokalt och från Internet. I båda fall måste koden vara sparad i en fil. Ska HTML-koden exekveras i en webbläsare måste filen som innehåller koden, ha ändelsen **html**, vare sig vi har lagrat filen lokalt eller hämtat den från en server på Internet. När vi testar våra koder i webb-utvecklingskursen gör vi det lokalt.

HTML – webbens standardspråk

HTML står för *HyperText Markup Language* och är webbens standardspråk. Syftet med HTML är att producera presentabla *dokument* som kombinerar text, bild och andra element. Koden genererar dokumentet som sedan visas på webben. Koden man skriver, är separerad från dokumentet – till skillnad från andra formateringsverktyg som t.ex. *Word*, där man direkt skriver i dokumentet. Man talar om att *Word* är ett s.k. **WYSIWYG**-verktyg, dvs *What You See Is What You Get*. I *Word* ser man, vad man *får* i dokumentet. Koden (VB Script) genereras automatiskt i bakgrunden och är osynlig för användaren.

Till skillnad från *Word* är HTML ett icke-**WYSIWYG**-verktyg, dvs man skriver koden utan att se dokumentet. Koden måste först tolkas av en interpretator, innan dokumentet kan uppstå. Webbläsaren som själv en programvara, är en sådan interpretator som exekverar koden och visar dokumentet. Webbläsaren har utvecklats för att bl.a. tolka HTML. Andra exempel på icke-**WYSIWYG**-verktyg är *TeX/LaTeX*, en applikation för typsättning och presentation av text, speciellt matematiska uppsatser.

En viktig egenskap av HTML är:

HTML är **inte** case sensitive (skiftlägeskänslig).

Dvs HTML skiljer inte på små och stora bokstäver.

Script vs. program

Med *script* menas all kod som, inbäddad i HTML, kan köras på webben inkl. HTML-kod själv. Och språk som används i en sådan kod kallas för *scriptspråk*. Exempel på *scriptspråk* är *JavaScript*, *PHP*, JavaScript används i regel på klientdatorer, medan PHP är webbservrarnas *scriptspråk*. Koder till *universella språk*, som kan användas för vilken applikation som helst och inte är begränsade till webben, kallas för *program*.

Det finns två olika kategorier av språk i datavärlden: *scriptspråk* och *universella språk* som t.ex. C, C++, C#, Java, Python, HTML är ett *scriptspråk* (Läs om script i nästa paragraf). HTML har även möjligheten att bädda in andra *scriptspråk* i sin kod som t.ex. JavaScript, förutsatt att man avgränsar språken genom tydliga markeringar. Webbläsaren är den naturliga exekveringsmiljön både för HTML och alla andra *scriptspråk*, medan särskilda verktyg behövs för att exekvera *universella språk*. För att *skriva* *script*kod behöver man endast en *editor*, och för att exekvera den en *webbläsare*. Vi nöjer oss med denna minimalistiska miljö vad gäller *scriptspråken*, för att förenkla den tekniska hanteringen och koncentrera oss på själva språket.

Det mest kända *scriptspråket* är *JavaScript* som skapades år 1995 av *Netscape*, ett amerikanskt mjukvaruföretag som 1994 lanserade den första grafiska webbläsaren *Mosaic* som snabbt blev en jättesuccé. Netscape integrerade JavaScript i *Mosaic*, för att göra webben interaktiv, se [De tre skikten](#) sid 8.

JavaScript får inte förväxlas med Java. Det handlar om två olika programmeringsspråk som dessutom tillhör två olika kategorier av programmeringsspråk: Medan JavaScript är ett *scriptspråk* är Java ett *universellt programmeringsspråk*.

Hos *scriptspråken* nöjer man sig med de enklare elementen i programmering, för att förse webbsidor med vissa funktionaliteter. Scripten bakas in i HTML-kod, varför de kan exekveras på webben.

Scriptspråkens *exekveringsmiljö* är webbläsaren (web browser).

Scriptspråken är *interpreterande språk*, dvs koden tolkas till maskinkod (datorns språk) av en interpretator som är inbyggd i webbläsaren. Maskinkoden utförs direkt av datorns processor utan att den mellanlagras. De mest använda webbläsarna är Google Chrome på Windwos-datorer och Safari på Mac-datorer. I båda är en interpretator för JavaScript inbyggd.

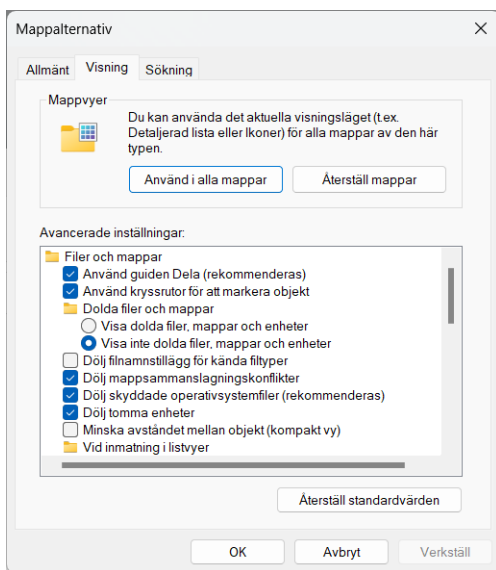
Utvecklingsmiljön däremot – dvs där man skriver koden – kan vara vilken editor som helst. Till skillnad från HTML är JavaScript-kod case sensitive, dvs JavaScript skiljer på små och stora bokstäver. Det gör inte HTML.

Filändelser

Skriver du din JavaScript kod i någon editor och sparar filen som ***.txt**, kommer du inte kunna exekvera den i en webbläsare, när du (dubbel)klickar på den. Boven i dramat är filändelsen: Operativsystemet identifierar de filer som innehåller kod via filändelsen. All JavaScript kod är inbakad i HTML kod, webbläsarens språk. Ska koden exekveras i en webbläsare måste filen som innehåller koden, ha ändelsen **html**, för att kunna identifieras som en JavaScript källkodsfil. Därför måste du aningen från början spara din källkodsfil med ändelsen **html** eller i efterhand ändra filändelsen till **html**. I Windows kallas filändelser för *Filnamnställäg*.

För att kunna följa reglerna för filändelsen som beskrevs ovan, förutsätts att man kan *se* filändelserna när man öppnar en mapp. Men i praktiken är detta ofta inte fallet. Orsaken är på operativsystemets inställningar. I Windows är default inställningen att man i regel *inte* kan se dem. Ta själv reda på hur det är på din dator. Så här kan man göra för att synliggöra filändelserna i Windows:

- Öppna en mapp i Windows.
- Gå i mappens menyrad till Mappalternativ. Om du inte hittar denna meny klicka på de tre små punkterna till höger (Visa mer) och välj Alternativ.
- Du borde få upp dialogrutan Mappalternativ. Välj fliken Visning. Bocka av rutan Dölj filnamnställäg för ända filtyper. Så här borde nu dialogrutan se ut:
- Klicka på knappen Använd i alla mappar, sedan på Ja och OK.



Nu borde du kunna se dina filers ändelser och kunna följa reglerna på förra sidan. Generellt rekommenderas att ha synliga filändelser på sin dator, när man programmerar.

1.3 Att komma igång med HTML

För att komma igång med HTML kan vi nu skriva våra koder i en valfri texteditor och spara filen som ren, dvs oformaterad textfil med ändelsen **html** (OBS! inte **txt**) på datorn. När vi sedan (dubbel)klickar på filen, kommer koden att exekveras i webbläsaren. Anledningen till det är att webbläsaren är ett program som kan tolka och exekvera **html**-kod: Webbläsaren är en **html**-interpretator. Så här kommer vi att testa alla våra HTML-koder i denna kurs. Även om man gör detta i en annan miljö är det i grund och botten denna teknik som används i bakgrunden.

Vårt första HTML-script

Öppna din favorit editor (t.ex. Notepad, Notepad++, TextPad i Windwos eller Textredigerare, TextEdit, emacs på Mac) och skriv följande kod. Ta bort radnumren **1-7** som har satts endast för att underlätta kodförklaringen. I övrigt bibehåll layouten:

```
1 <!-- Welcome.html
2     Skriver ut en rad text i fet stil & storleken h1 -->
3
4 <head>
5     <title>Vårt första HTML-script</title>
6 </head>
7
8 <body>
9     <h1>Välkommen till HTML!</h1>
10 </body>
```

Spara koden i filen **Welcme.html**. (Dubbel)klicka på filen på den plats du sparar den. Din webbläsare kommer att visa körresultatet. Så här ser resultatet ut:



Vi kommer i fortsättningen att referera till koden som *scriptet* **Welcme** och till filen som **Welcme.html**. I följande går vi igenom koden i detalj.

Kommentarer

Raderna 1-2 i scriptet **welcome** är *kommentar*. dvs de utförs inte. De ska förklara koden. Allt som skrivs mellan `<!--` och `-->` betyder kommentar, HTML koden `<!--` inleder och `-->` avslutar en kommentar som kan sträcka sig över flera rader eller stå mitt på en rad. Att skriva kommentarer tillhör god programmeringsstil. Även indragningarna, på eng. *indentations*, på raderna 5 och 9 är element av god programmeringsstil som ska hjälpa oss att förstå att den indragna koden är nästlad i en övergripande struktur som inte är indragen, när vi läser scriptet.

HTML-taggar

En *tagg* i HTML inleds med `<` och avslutas med `>`. Taggar används i regel parvis: en *starttagg* markerar *början* och en *sluttagg* markerar *slutet*. På rad 9 har vi:

```
<h1>Välkommen till HTML!</h1>
```

Denna rad har två taggar: starttaggen `<h1>` och sluttaggen `</h1>`.

Det som står mellan start- och sluttagg, texten **Välkommen till HTML!**, kallas för *innehåll*, närmare bestämt innehåll till `<h1>`-elementet (se nästa rubrik).

Taggar är HTML-språkets minsta byggstenar. HTML är ett s.k. *taggat märkspråk*. Det som skrivs *inom* en tagg, dvs mellan `<` och `>`, är HTML-kod. Allt som står *utanför* taggen kommer att visas som text på webbsidan, se regeln på nästa sida.

Taggar kan nästlas i varandra.

HTML-element

Ett HTML-element har följande ingredienser:

Starttagg + innehåll + sluttagg = **Element**

`<h1>Välkommen till HTML!</h1>` är i sin helhet ett HTML-element, närmare bestämt ett `<h1>`-element. Element kan nästlas i varandra. I scriptet **welcome** är `<h1>`-element nästlat i ett `body`-element (raderna 8-10).

Ett annat exempel på element hittar vi på rad 5:

```
<title>Vårt första HTML script</title>
```

Även detta element är nästlat i ett annat element, nämligen i `head`-elementet på raderna 4-6.

Om taggar är HTML-språkets minsta byggstenar, så är element språkets näst minsta byggsten.

En övergripande struktur

`head`- och `body`-elementen i scriptet `Welcome` kallas för *struktureringselement*. De delar in scriptet `Welcome` i två skilda delar: ett huvud och en kropp. Medan i kroppen kodas webbsidans huvudinnehåll, är huvudet tänkt för att innehålla bl.a. `title`-elementet, men även andra ingredienser som vi kommer att lära känna senare. Alla HTML-script följer den här övergripande `head`- och `body`-strukturen.

På raderna **4-6** står `head`-elementet som nästlar `title`-elementet. All text som skrivs som innehåll i `title`-elementet på rad **5** kommer att synas på rubriken av webbläsarens flik längst upp till vänster, kallad *title bar*, när man kör scriptet. Det är texten **Vårt första HTML-script** som kan ses där (lite svåräst). Utelämnar man `title`-taggen kommer scriptets filnamn att skrivas i title bar.

På raderna **8-10** står `body`-elementet som nästlar `h1`-elementet på rad **9** som omgärdar texten som ska skrivas ut: **Välkommen till HTML!**. `h1`-taggen bestämmer textens stil och storlek: Texten skrivs ut i fet stil och i den storlek som man ser i körresultatet på förra sidan. Med `h`-taggar kan man åstadkomma rubriker i texten. `h` står för *header*, dvs rubrik. Det finns flera header-taggar i HTML som vi kommer att lära känna i nästa avsnitt. `h1` är den största av dem.

Observera att texten som skrivs ut, inte behöver sättas inom varken apostrofer eller citationstecken i koden. Detta är en konsekvens av den regel som nämndes inledningsvis och som vi nu sammanfattar som en övergripande regel:

Inom en tagg står **HTML-kod**. Allt som står utanför taggar anses vara **text** som ska visas på webbsidan.

Denna regel återspeglar HTMLs karaktär som ett icke-WYSIWYG-verktyg (sid 12).

Headers

```
1 <!-- Headers.html
2     Skriver ut flera rader text i olika stilar och storlekar -->
3 <head>
4     <title>Olika stilar och storlekar för rubriker</title>
5 </head>
6 <body>
7     <h1> Rubrik i HTML med h1 </h1>
8     <h2> Rubrik i HTML med h2 </h2>
9     <h3> Rubrik i HTML med h3 </h3>
10    <h4> Rubrik i HTML med h4 </h4>
11    <h5> Rubrik i HTML med h5 </h5>
12    <h6> Rubrik i HTML med h6 </h6>
13 </body>
```

Öppna din favorit editor, skriv koden ovan och spara den i filen **Headers.html**. (Dubbel)klicka på filen när du sparat den. Webbläsaren visar:



Vi kommer att referera i fortsättningen till koden ovan som *scriptet* **Headers**. Som man ser skriver scriptet **Headers** ut fyra rubriker i olika storlekar, förorsakat av HTMLs <h1>-taggar (i = 1, 2, 3, 4, 5, 6) som formaterar textens storlek.

En fråga dyker upp här: Vad har förorsakat radbyten i körresultatet ovan? Vi har inte skrivit i scriptet någon kod som ska åstadkomma radbyten. Slutsatsen är:

<h1>-taggarna (i = 1, 2, 3, 4, 5, 6) avslutar varje rubrik med radbyte.

Utskrift i flera rader

Frågan som automatiskt dyker upp efter förra avsnitts slutsats, är: Hur kan vi själva åstadkomma radbyten. Med radbryte menar vi förstås inte radbryte i koden utan i *utskriften*, dvs i körresultatet. Scriptet **Break** nedan visar *ett* sätt att i HTML åstadkomma egna radbyten och kombinerar detta med resultatet från förra avsnitt:

```
1 <!-- Break.html
2     Radbrytning i utskriften med HTMLs break-tagga <br> och
3     genom byte av h-taggens rubrikstil -->
4
5 <head>
6     <title>Utskrift i flera rader</title>
7 </head>
8
9 <body>
10    <h1>Välkommen till<br>Webbutveckling 1</h1><h2>med HTML och CSS!</h2>
11 </body>
```

Det första radbytet kodas med HTML-taggen `
` som bakas in i texten. Det andra radbytet genereras automatiskt genom att vi stänger `<h1>`-taggen och öppnar `<h2>`-taggen, se slutsatsen på förra sidan.

Körresultatet blir en utskrift på tre rader:



br-taggen

Scriptet **Break** använder HTML-taggen `
`, även kallad *break-taggen* genom att baka in den i texten **Välkommen till**`
`**Webbutveckling 1** (rad 8) för att åstadkomma radbrytning i utskriften. *break-taggen* kan placeras var som helst.

Man kan koda taggen med `
` eller med `
`, så att även sluttaggen syns. Det är lite smaksak. Anledningen är att **br** bildar ett s.k. *tomt element*, som har vissa regler, vilket förklaras senare (sid 27). Vi nöjer oss än så länge med `
` för enkelhetens skull.

- 1.1 Vad är *embedded systems* och hur skiljer de sig från *Internet of Things (IoT)*?
- 1.2 Hur tolkar du termen *artificiell intelligens*? Kan maskiner ”tänka”?
- 1.3 Varför tror *Steve Jobs* att programmering lär oss hur vi ska *tänka*?
- 1.4 Vad är relationen mellan webbutveckling och programmering?
- 1.5 Vilka verktyg behöver man för webbutveckling?
- 1.6 Nämn webbutvecklingens tre skikt. Varför ska man skilja dem från varandra?
- 1.7 Vad menas med programmeringstänkande?
- 1.8 Hur skulle du definiera begreppet *algoritm*?
- 1.9 Är datorprogram ett sätt att *beskriva* algoritm? Om ja, är det det enda sättet?
- 1.10 Försök att med egna ord beskriva *algoritmiskt tänkande*.
- 1.11 På vilket sätt kan man visualisera en algoritms logiska struktur?
- 1.12 Vad har algoritmiskt tänkande med programmering att göra?
- 1.13 Vad innebär *modularisering* och varför är den relevant för programmering?
- 1.14 Använder du i vardagen algoritmer? Om ja, nämn några exempel.
- 1.15 Vilka är algoritmers byggstenar?
- 1.16 Vad är skillnaden mellan *instruktioner* och *val* i en algoritm?
- 1.17 Vad är kontrollstrukturer? Nämn tre exempel på dem.
- 1.18 Varför kan man inte lära sig webbutveckling genom att endast läsa böcker?
- 1.19 Vilken egenskap borde editorn ha i vilken man skriver kod?
- 1.20 Vad är en IDE? Vad är skillnaden till en editor?
- 1.21 Vad är en webbläsare?
- 1.22 Vad är en *interpretator* och hur skiljer den sig från en *kompilator*?
- 1.23 Vad innebär *kompilering* och hur skiljer den sig från *exekvering*?
- 1.24 Skriver man *källkod* eller *maskinkod* när man kodar i något språk?

- 1.25 Vilka två kategorier av språk är relevanta för webbutveckling?
- 1.26 Är HTML ett *universellt* programmeringsspråk?
- 1.27 Är HTML ett interpreterande eller ett kompilerande språk?
- 1.28 Är HTML case sensitive?
- 1.29 Vad är ett *script* och hur skiljer det sig från ett *program*?
- 1.30 Nämn några exempel för *scriptspråk* och några för *universella* språk.
- 1.31 I vilken miljö exekveras HTML-kod?
- 1.32 Vad är ett WYSIWYG-verktyg? Nämn några exempel.
- 1.33 Är HTML ett WYSIWYG-verktyg?
- 1.34 Vilka verktyg behöver man för att kunna utveckla HTML-script?
- 1.35 Vilka typer av ordbehandlingsprogram är olämpliga för webbutveckling?
- 1.36 Varför är filändelser relevanta för en webbutvecklare?
- 1.37 Hur kodar man kommentar i HTML?
- 1.38 Varför är kommentarer viktiga när man skriver kod?
- 1.39 Vilken övergripande struktur bör man följa i alla HTML-script?
- 1.40 Var någonstans i dokumentet hamnar texten man skriver i **<title>**-taggen?
- 1.41 Vilken text kommer att skrivas om man utelämnar **<title>**-taggen?
- 1.42 Nämn några exempel på *headers* (rubriktaggar) i HTML. Hur många finns?
- 1.43 Vilka egenskaper får en text som formateras av **<h1>**-taggen?
- 1.44 Vad är en bieffekt av headers som inte syns i koden?
- 1.45 Nämn några exempel på HTML-taggar som saknar sluttagg.
- 1.46 Vad består ett HTML-element av?
- 1.47 Vad menas med ett *tomt* HTML-element?
- 1.48 Hur kodar man radbyte i HTML?
- 1.49 Nämn två olika sätt att koda break-taggen.
- 1.50 Varför bildar break-taggen ett tomt element?

- 1.1 Om du har en favorit editor, öppna den (sid 11). Om inte, ladda ned en editor (t.ex. Notepad++ eller Emacs) och installera den. Undersök i editorn skillnaderna – vad gäller formen och utseendet – mellan tecknen *apostrof* ('), *citationstecken* ("), *accent* (´), *slash* (/) och *backslash* (\), så att du kan skilja dem från varandra. Försök att memorera deras tangenter på din dators tangentbord.
- 1.2 Visar din dator filändelserna när du öppnar en mapp? Om inte, genomför instruktionerna **Filändelser** på sid 14, för att synliggöra filändelserna.
- 1.3 Öppna din favorit editor och mata in koden till scriptet **Welcome** (sid 15). Bibehåll layouten. Spara koden i filen **Welcome.html**. (Dubbel)klicka på filen, så att den körs i din webbläsare.
- 1.4 Modifiera scriptet **Welcome** genom att ändra texten i <title>-taggen till ditt namn och texten som skrivs ut i dokumentet, till: **Det här scriptet har jag skrivit själv!** Spara koden i filen **Mitt.html** och kör den i din webbläsare. Välj en lämplig mapp på din dator för att spara filen. Skapa en mappstruktur för scriptfiler du kommer att ha användning av för kursens kodexempel.
- 1.5 Modifiera koden i scriptet **Headers** (sid 17), så att de fyra utkriftsraderna syns i växande textstorlekar istället för minskande.
- 1.6 Skriv ett HTML-script som åstadkommer följande utskrift:

```
*
**
***
****
*****
*****
```
- 1.7 Skriv kod som ger följande utskrift:

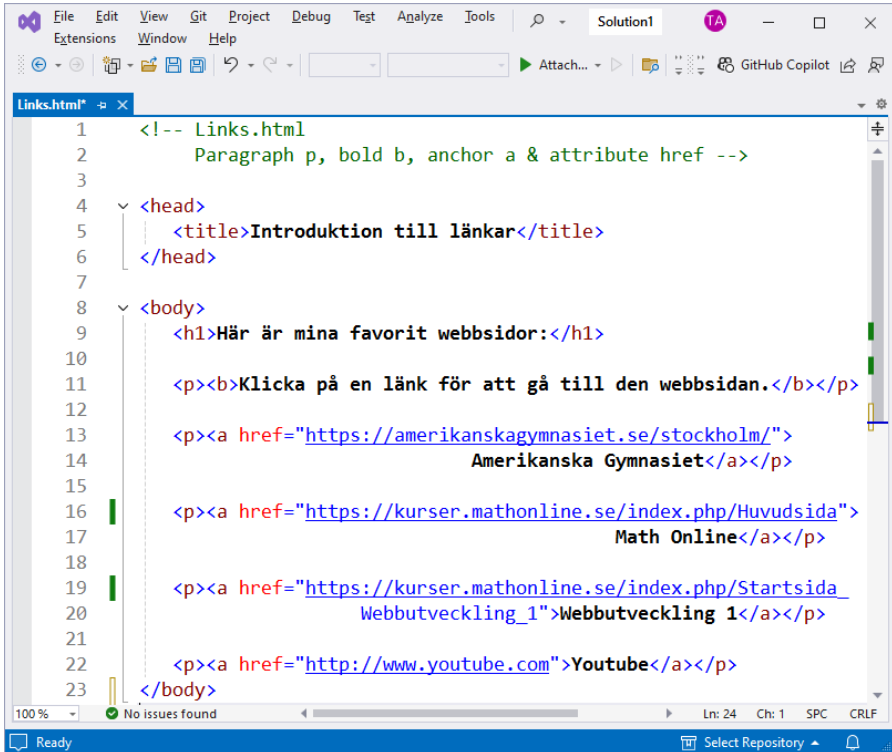
```
****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
```

Kapitel 2

Grundbegrepp i webbutveckling

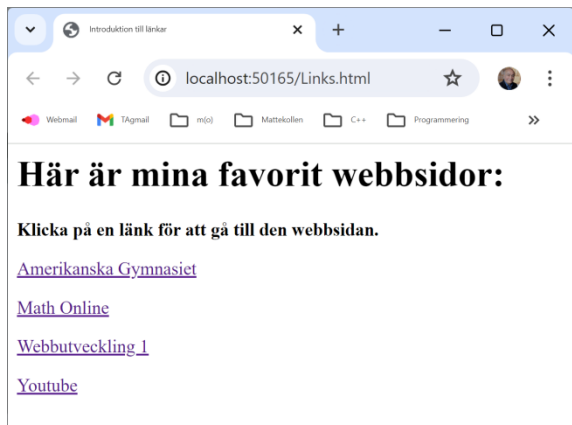
Ämne	Sida	Program
2.1 Länkar	24	Links
- Elementet ankare	25	Contact
- Attribut	26	
- Varför heter det ankare?	26	
2.2 Bilder	27	Picture
- img – ett tomt element	27	
- Förkortningsregeln för tomma element	28	
- Val av bildstorlek	29	
2.3 Bilder som länkar	30	Nav
- Nästlade element	31	
- Navigeringsmeny	31	
- Interaktion	32	
2.4 Specialtecken	33	Contact2
- Namnkoder för specialtecken	33	
- Talkoder för specialtecken	34	
- hr-taggen	35	
2.5 Punktlister	36	Links2
2.6 Nästlade och ordnade listor	38	List
Frågor till kap 2	41	
Övningar till kap 2	44	

2.1 Länkar



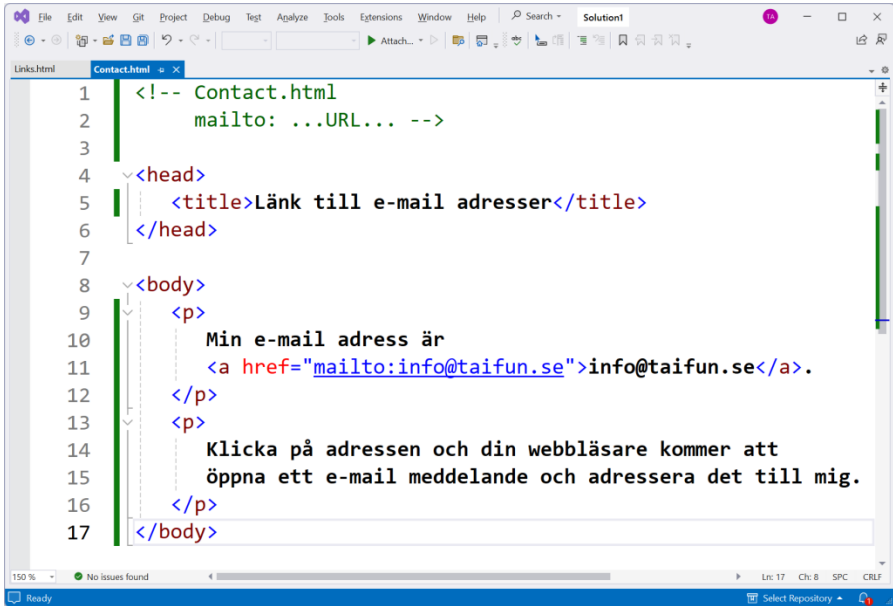
```
1 <!-- Links.html
2     Paragraph p, bold b, anchor a & attribute href -->
3
4 <head>
5     <title>Introduktion till länkar</title>
6 </head>
7
8 <body>
9     <h1>Här är mina favorit webbsidor:</h1>
10
11     <p><b>Klicka på en länk för att gå till den webbsidan.</b></p>
12
13     <p><a href="https://amerikanskagymnasiet.se/stockholm/">
14         Amerikanska Gymnasiet</a></p>
15
16     <p><a href="https://kurser.mathonline.se/index.php/Huvudsida">
17         Math Online</a></p>
18
19     <p><a href="https://kurser.mathonline.se/index.php/Startsida
20         Webbutveckling_1">Webbutveckling 1</a></p>
21
22     <p><a href="http://www.youtube.com">Youtube</a></p>
23 </body>
```

Scriptet `Links` introducerar elementtyperna *paragraph* `p` för stycke, *bold* `b` för fet stil och *anchor* `a` för *anchor*. `p` skapar ett nytt stycke (paragraf) i textflödet. `b` byter textens stil till fet. Ankaret `a` skapar en länk. `a`'s *attribut* `href` specificerar webbadressen. Så här ser korrespondansen ut där man kan navigera till de specificerade webbsidorna:



Attributnamnet `href` refererar till attributvärdet "http: ... (URL) ..." som skrivs inom citationstecken. URL står för `Uniform Resource Locator` och är formatet för giltiga webbadresser på Internet.

Låter man inleda attributvärdet istället för `http` med `mailto` kan man på samma sätt länka till mailadresser: "`mailto: ... (mailadress) ...`". Scriptet **Contact** demonstrerar detta:



```
1 <!-- Contact.html
2     mailto: ...URL... -->
3
4 <head>
5     <title>Länk till e-mail adresser</title>
6 </head>
7
8 <body>
9     <p>
10        Min e-mail adress är
11        <a href="mailto:info@taifun.se">info@taifun.se</a>.
12    </p>
13    <p>
14        Klicka på adressen och din webbläsare kommer att
15        öppna ett e-mail meddelande och adressera det till mig.
16    </p>
17 </body>
```

Så här ser körresultatet av scriptet **Contact** ut:



Elementet ankare

Den generella formen på HTML-elementet *ankare* (eng. *anchor*) är:

```
<a href="mailto:emailadress">...text/bild/emailadress...</a>
```

där `...text/bild...` är innehållet i elementet som blir en klickbar länk när man kör scriptet. Raden **11** i scriptet **Contact** ovan är ett exempel på denna form. Där skapas en länk till e-mail. `href` är ankarets attribut, närmare bestämt dess *namn*, medan "`mailto:emailadress`" är attributets *värde*.

Attribut

Observera att attributnamnet **href** samt dess värde är en del av ankar-elementets *starttagg*, inte av innehållet. Dvs det skrivs innan man avslutar starttaggen med **>** . Så här ser starttaggen ut i sin helhet, dvs med attributet:

```
<a href="mailto:emailadress">.
```

Att attributet inte är del av innehållet utan av starttaggen, beror på den regel som vi tidigare lärde oss om HTMLs övergripande strukturer (sid 17):

Inom en tagg står **HTML-kod**. Allt som står utanför taggar anses vara **text** som ska visas på webbsidan.

Attributet **href** är HTML-kod. Därför måste det stå *inuti* ankarens starttagg. Det är innehållet **...text/bild...** som ska visas på webbsidan. Därför måste det stå *utanför* taggarna, närmare bestämt mellan ankarens start- och starttagg, se raden **11** i scriptet **Contact** på förra sidan eller raderna **13-22** i scriptet **Links** (sid 24).

Varför heter det ankare?

Ett ankare anses vara en *förankringspunkt* för en förbindelse mellan två platser på webben: vårt HTML-dokument som vi aktuellt kör och webbsidan som Internet-platsen (URLen) i attributet **href** länkar till. Det är även möjligt att man länkar till samma HTML-dokument, men på en annan plats i dokumentet. Elementtypen **a** som står för engelskans *anchor* är den fasta punkten – *ankaret* – för dessa två platser. Den binder ihop scriptet med dokumentet.

2.2 Bilder

Hittills ha vi använt endast text i våra scriptexempel. Men en webbsida innehåller i regel text *och* bild. Att formge webbsidor med bilder är en väsentlig del av webbdesign. Bilder lagras i datorn i ett visst *format*, vilket framgår av filändelsen.

De mest använda bildformat på webben är *Joint Photographic Experts Group* (JPG / JPEG) och *Graphics Interchange Format* (GIF). För att hantera bilder på webben rekommenderas, som det sades i början (sid 7), att ha ett bildbehandlingsprogram, t.ex. *Photoshop* eller motsvarande. Vi kommer att ha nytta av det i detta avsnitt, då vi kommer att behöva ta reda på t.ex. bildernas storlek, deras upplösning och ev. att anpassa deras storlek till våra varierande behov på webben. Följande script introducerar oss till att använda bilder i HTML:

```
1 <!-- Picture.html
2     img-taggen med attributen src och alt
3     samt height & width -->
4
5 <head>
6     <title>Bilder i HTML</title>
7 </head>
8
9 <body>
10     <p>
11         
13         
15     </p>
16 </body>
```

img – ett tomt element


Raderna 11-12 använder **img**-elementet för att placera bilder i dokumentet. Den generella formen på ett **img**-element är:

```

```

Anledningen till att det är tomt, är att det egentligen borde se ut så här:

```
/img>
```


Och mellan starttaggen `` och sluttaggen `` finns inget innehåll , därför är det tomt. Elementet ska inte skriva ut text, utan endast infoga en bild.

Förkortningsregeln för tomma element

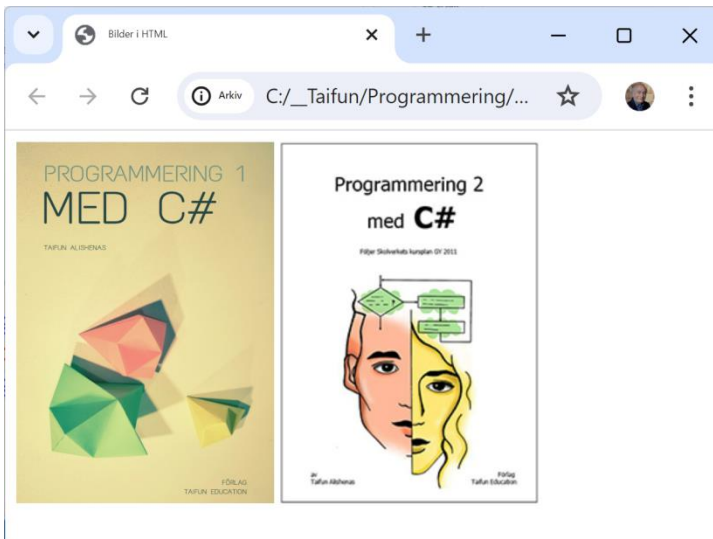
Är ett element tomt kan man tillämpa följande regel i HTML för att förkorta koden:

Ett **tomt element** i HTML kan avslutas på tre sätt:

1. med sluttaggen: `<starttag></sluttagg>` eller
2. endast med `/>`: `<starttag />` eller
3. endast med `>`: `<starttag>`

 är det tomma innehållet. Punkt 1 visar det vanliga sättet att avsluta elementet: t.ex. `img`-elementet kan avslutas på vanligt sätt med ``. Punkt 2 är ett sätt att förkorta koden: som en förenkling kan man avsluta endast med `/>`. Vi har i scriptet `Picture` på raderna 12 och 14 valt att använda den förenklade varianten 2. Mellan-slaget före `/>` har lagts in endast för bättre läslighetens skull. Punkt 3 är ännu kortare: endast med `>`. Dvs bara starttaggen står där fullständigt. I praktiken har dock starttaggen ofta ett antal attribut, innan den avslutas, som vi utelämnat här. Vi har använt punkt 3 i scriptet `Nav`: s `img`-element på sid 30.

Annars gör `img`-elementet även med de förenklade varianterna exakt samma sak som med den vanliga, nämligen att infoga en bild i dokumentet. Testa gärna båda varianterna. Så här ser körresultatet av scriptet `Picture` ut:



Bildernas **jpg**-filer som är värden till attributet **src** (source) måste ligga i samma mapp som scriptfilen **Picture.html**, för att webbläsaren ska kunna ladda dem i dokumentet. Ligger de inte där eller kan webbläsaren av andra skäl inte ladda dem, kommer texten som är tilldelad attributet **alt** (alternative) att skrivas ut istället. Testa gärna!

Val av bildstorlek

Bildernas storlek i dokumentet bestäms i koden av **img**-elementets attribut **height** och **width**. Enheten till storleken är *pixlar (picture elements)*. Tittar man på bildernas originalstorlek i resp. **jpg**-fil – antingen i filens Egenskaper → Information (i Windows) eller i Photoshop (Image → Image Size) hittar man förstås andra värden. Hur ska man anpassa dem till dokumentet? Man bildar kvoten höjd / bredd av bildernas originalstorlek. I vårt fall får man ca. **1,4**. För att behålla bildernas rätta proportion i HTML-dokumentet, måste denna kvot tas över till koden. Dvs **img**-elementets attribut **height** och **width** måste väljas så att även $\text{height} / \text{width} = 1,4$. Väljer man andra värden tappar bilderna sina ursprungliga proportioner och blir förvrängda i dokumentet, vilket inte kommer att se bra ut. Utelämnar man attributen **height** och **width** kommer webbläsaren välja bildernas originalstorlek.

2.3 Bilder som länkar

```
File Edit View Git Project Debug Test Analyze Solution1
Tools Extensions Window Help
Nav.html* x
1 <!-- Nav.html
2     Klickbara bilder med img-element, nästlade
3     i a-element (ankare) som kör HTML-script -->
4 <head>
5     <title>Navigeringsmeny</title>
6 </head>
7 <body>
8     <a href="Old/Welcome.html">
9         
11     </a><br>
12
13     <a href="Old/Headers.html">
14         
16     </a><br>
17
18     <a href="Old/Break.html">
19         
21     </a><br>
22
23     <a href="Links.html">
24         
26     </a><br>
27
28     <a href="Contact.html">
29         
31     </a><br>
32
33     <a href="Picture.html">
34         
36     </a><br>
37 </body>
100% No issues found Ln: 38 Ch: 1 SPC CRLF
Ready Select Repository
```

Nästlade element

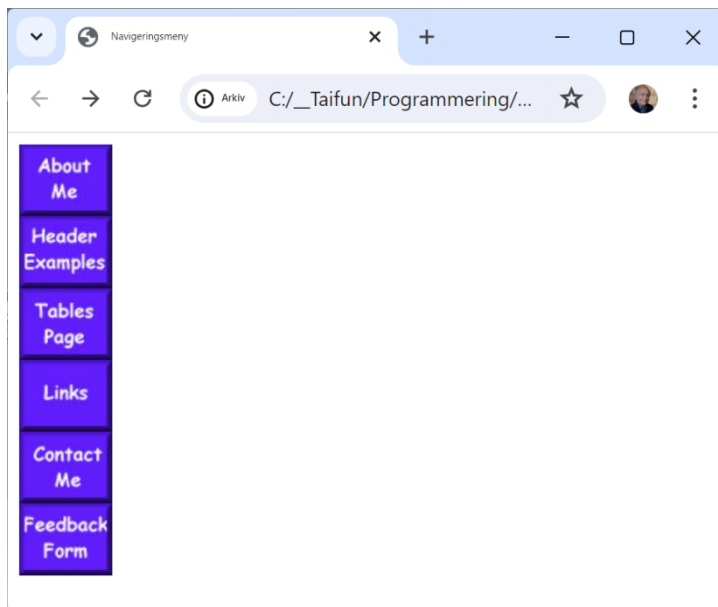
Scriptet **Nav** nästlar **img**-elementet i **a**-elementet (ankare), dvs sätter in **img** i ankarets innehåll. Det gör det 6 gånger, första gången på raderna **8-11**. Kör man **Nav** visas 6 små bilder. Deras samling kallar vi för Navigeringsmeny i title-taggen. Bilderna hämtas från undermappen **buttons** som ligger i samma mapp som scriptfilen **Nav.html**, eftersom **buttons** är den mapp där vi har placerat alla bildfiler av typ ***.jpg**. Det är **img** som visar dem på raderna **9, 14, 19, 24, 29** och **34**.

P.g.a. nästlingen i ankaret **a** blir de 6 små bilderna klickbara. Dvs de länkar till de webbsidor som **a**-elementets attribut **href** refererar till. T.ex. hämtar **img**-elementet bildfilen **Form.jpg** från mappen **buttons** till dokumentet (raderna **34-35**). Genom att **img** är nästlat som innehåll i ankaret **a**, blir bilden en länk som leder till **a**:s attribut **Picture.html** (rad **33**). Dvs scriptet **Picture:s** körresultat, de två bokomslagen, visas. Se nästa sida.

Alla andra delar av scriptet **Nav** fungerar på samma sätt. Tre scriptfiler av typ ***.html** ligger i undermappen **Old** (raderna **8, 13, 18**). De andra finns direkt i samma mapp som scriptfilen **Nav.html**.

Navigeringsmeny

Så här ser ett körresultat av scriptet **Nav** ut:



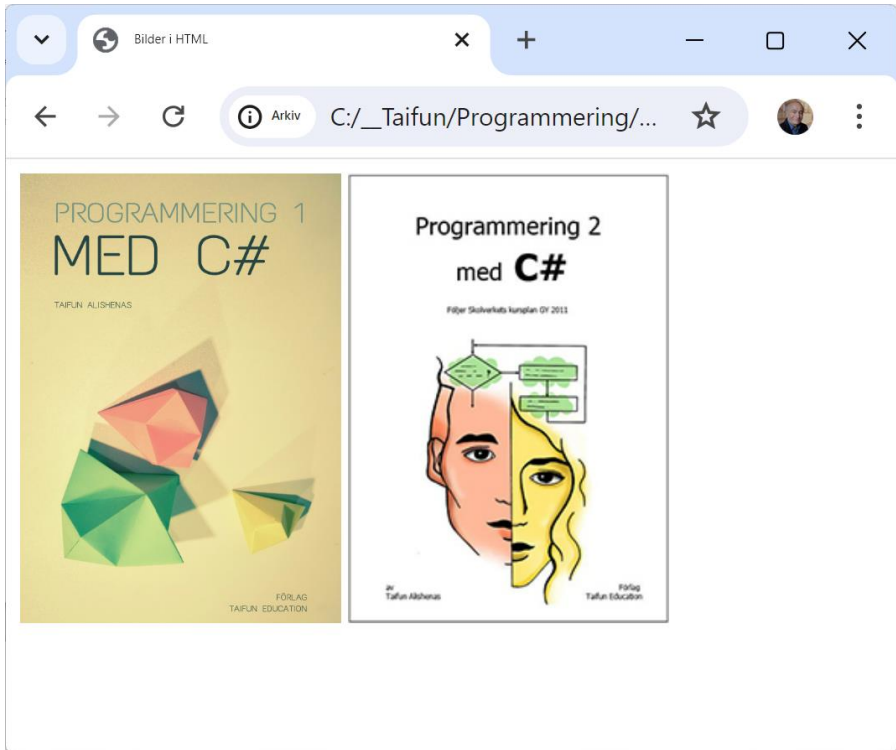
Sex små bilder visas som hämtats med **img**-elementet. Bilderna bildar en slags meny för navigering till andra hemsidor vars länkar finns bakom bilderna. Implemen-

terat är detta i scriptet **Picture** med **img**-element nästlade i **a**-element, vilket gör att bilderna blir klickbara. För varje bild används samma konstruktion.

Texterna på bilderna är lite godtyckliga och har mindre betydelse. Se över dem. Man kan lika bra använda andra bilder. Testa gärna!

Interaktion

Men så länge vi inte interagerar med dokumentet händer ingenting. Klickar vi däremot i navigeringsmenyn ovan på den sista bilden som har texten **Feedback Form**, visas dokumentet som tidigare kodats i scriptet **Picture** (sid 27):



Vi klickade i navigeringsmenyn (förra sidan) på den sista bilden som hade texten **Feedback Form**. Eftersom denna bild enligt rad 36 är lagrad i filen **buttons/-Form.jpg** och enligt rad 35 länkar till filen **Picture.html**, blir det samma bild (sid 28) som tidigare producerats av scriptet **Picture**.

Denna kod producerar tecknet > . Rad 14 i scriptet ovan använder den.

Ett annat exempel för definitionen ovan är specialtecknet &, kallat *ampersand*, som enligt definition inleder koden till just de reserverade specialtecknen. För att skriva ut själva tecknet & måste man koda &

När man kodar specialtecken följer efter ampersand ett *namn*, i exemplen: **amp**, **quot**, **apos**, **nbsp**. Alla koder avslutas med semikolon. T.ex. producerar koden **&frac13**; bråket 1/3. Men tyvärr är koden inte generell, dvs andra bråk kan inte bildas på liknande sätt.

Andra, mer eller mindre intressanta specialtecken ges exempel på i scriptet **Contact2** på förra sidan vars körresultat ser ut så här:



Talkoder för specialtecken

Det finns möjligheten att ange *talkoder* istället för *namn*, när man kodar tecken i HTML, inkl. specialtecken. T.ex. ger **&**; samma som **&**; nämligen tecknet &, där 38 är ASCII-koden till tecknet &. ASCII är den äldsta och mest använda teckenstandarden och en del av, närmare bestämt, början av *Unicode*-tabellen. Faktiskt kan även vanliga tecken, inte bara specialtecknen, kodas med denna metod, dvs:

&#Unicode;

Unicode's **0-127** kallas för ASCII-koder. I övn 2.5 kan du öva dig på talkoder. Där hittar du även fler exempel på talkoder (sid 44).

En bra översikt över specialtecken med massor av nyttig information ger Internet-länken:

<https://www.enur.se/tecken.html>

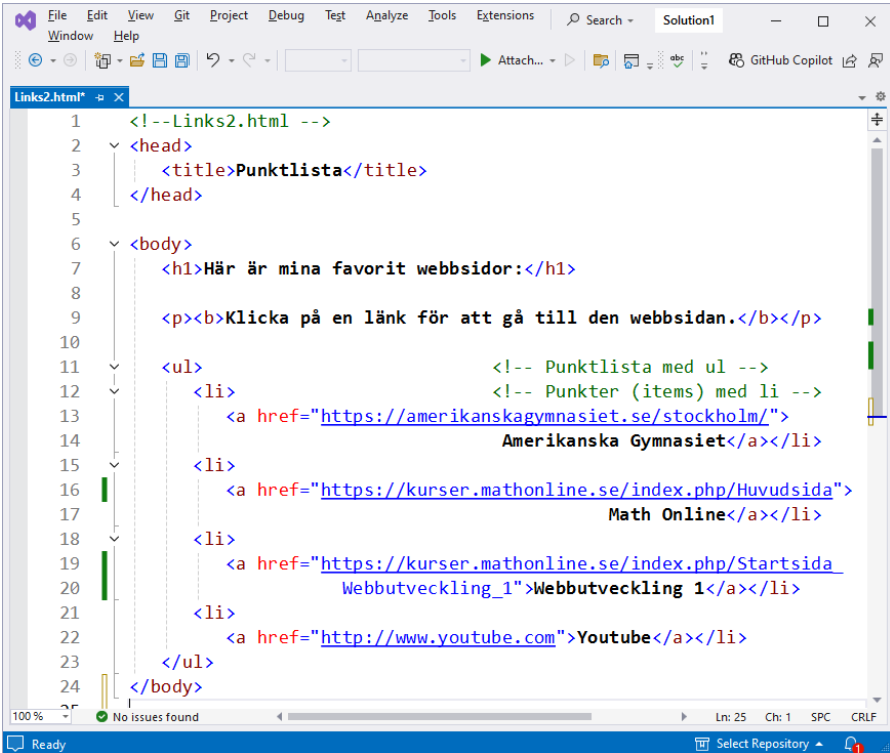
hr-taggen

Raden **18** i scriptet på förra sidan har inget med specialtecken att göra, utan introducerar **hr**-taggen (*horizontal rule*) som drar en horisontell linje i dokumentet, vilket kan beskådas i körresultatet ovan.

Man kan koda denna tagg med `<hr>` eller med `<hr />`. Det är lite smaksak. Anledningen är att **hr** bildar ett tomt element, för vilket gäller förkortningsregeln som nämndes tidigare (sid 28), när vi introducerade det tomma **img**-elementet (sid 27).

Samma sak är det med **br**-taggen: `
` eller `
` (sid 19).

2.5 Punktlistor



```
1 <!--Links2.html -->
2 <head>
3   <title>Punktlista</title>
4 </head>
5
6 <body>
7   <h1>Här är mina favorit webbsidor:</h1>
8
9   <p><b>Klicka på en länk för att gå till den webbsidan.</b></p>
10
11  <ul>                                <!-- Punktlista med ul -->
12    <li>                                <!-- Punkter (items) med li -->
13      <a href="https://amerikanskagymnasiet.se/stockholm/">
14        Amerikanska Gymnasiet</a></li>
15
16      <a href="https://kurser.mathonline.se/index.php/Huvudsida">
17        Math Online</a></li>
18
19      <a href="https://kurser.mathonline.se/index.php/Startsida
20        Webbutveckling_1">Webbutveckling 1</a></li>
21
22      <a href="http://www.youtube.com">Youtube</a></li>
23    </ul>
24 </body>
```

Elementet unordered list **ul**

Scriptet **Links2** (ovan) introducerar elementtypen *unordered list* **ul** för punktlista. Listan är *unordered* (oordnad) eftersom den inte ställer upp sina items med ordnande symboler, typ bokstäver eller siffror, utan med punkter.

- Point A
- Point B
- Point C

ul förser sina items med s.k. *bullets* (punkter). Därför betecknas oordnade listor även som *bulleted lists* (punktlistor). I scriptet **Links2** sträcker sig **ul**-elementet över raderna **11-23**.

li-taggen

Dessvärre genererar **ul** sina items inte automatiskt. De måste kodas med en speciell tagg: **li**-taggen som står för *list item*. I scriptet **Links2** finns **li**-taggen på raderna **12, 15, 18** och **21**. Varje **li**-tagg i sin tur inleder ett element vars innehåll är en ankare.

Jämför man vårt aktuella script **Links2** med ett tidigare script **Links** (sid 24) kan man se att samma länkar där finns nu i en punktlista **ul** och att paragraferna där har bytts ut mot punktlistans items som kodas med **li**-taggen.

Körresultatet ser ut så här. De tidigare webblänkarna från scriptet **Links** finns nu i en punktlista:



2.6 Nästlade och ordnade listor

```
1 <!-- List.html -->
2 <head>
3   <title>Nästlade och ordnade listor</title>
4 </head>
5 <body>
6   <h1>Internets bästa egenskaper:</h1>
7   <ul> <!-- Punktlista -->
8     <li>Du kan träffa folk från hela världen</li>
9     <li>
10      Du får tillgång till nya media så snart de publiceras:
11     <ul> <!-- Nästlad punktlista med annorlunda punktsymbol -->
12       <li>Nya spel</li>
13       <li>
14         Nya applikationer:
15         <ol type = "I"> <!-- Ordnad nästlad lista -->
16           <li>för jobb</li> <!-- med attribut type som -->
17           <li>för nöje</li> <!-- bestämmer listsymbolen -->
18         </ol>
19       </li>
20       <li>Aktuella nyheter</li>
21       <li>Sökmaskiner</li>
22       <li>Shopping</li>
23       <li>
24         Programmering:
25         <ol type = "a"> <!-- Ny ordnad nästlad lista -->
26           <li>HTML</li> <!-- type bestämmer symbolen -->
27           <li>Java</li>
28           <li>Python</li>
29         </ol>
30       </li>
31     </ul> <!-- Avslutar nästlad punktlista från rad 11 -->
32   </li>
33   <li>Länkar</li>
34   <li>Hålla kontakt med gamla kompisar</li>
35   <li>Rapportera revolutionära händelser från diktatoriska länder</li>
36 </ul> <!-- Avslutar punktlista från rad 7 -->
37 <h1>Mina tre favorit språk:</h1>
38 <ol> <!-- Ordnad lista utan attributet type -->
39   <li>HTML</li> <!-- får en numerisk sekvens 1, 2, 3, ... -->
40   <li>CSS</li>
41   <li>JavaScript</li>
42 </ol>
43 </body>
```

Elementet ordered list **ol**

Scriptet **List** (ovan) introducerar elementtypen *ordered list* **ol**. Listan är *ordered* (ordnad) eftersom den ställer upp sina items med ordnande symboler, t.ex. bokstäver eller siffror, se bilden till höger. I scriptet börjar den första ordnade listan på rad **15** med elementtypen **ol**.

List of Fruits

1. Apple
2. Mango
3. Banana
4. Grapes
5. Orange

ol-attributet `type`

Elementtypen `ol` på rad **15** använder sitt attribut `type` för att bestämma listsymbolen. `type` har satts till "I", vilket betyder det romerska talet 1, så att listan börjar med romerskt I och fortsätter automatiskt med nästa romerska tal II. Körresultatet på nästa sida visar utseendet.

På rad **25** börjar den andra ordnade listan som sätter `type` till "a", så att listsymbolen blir det lilla latinska alfabetet. Den sista ordnade listan har inget `type`-attribut, så att den numeriska sekvensen 1, 2, 3, ... används by default. Varje nästlad lista blir automatiskt indragen, så att den hierarchiska strukturen blir tydlig.

Nästlade listor

Scriptet `List` (förra sid) innehåller ett antal nästlingar, dvs olika listtyper är inblandade och nästlade i varandra. De åstadkommer utskriften på nästa sida som återspeglar nästlingarna med olika nivåer av indragningar från vänstra kanten.

Om vi börjar titta på den överordnade strukturen i scriptet `List` kan vi konstatera att en oordnad punktlista `ul` börjar på rad **7** och slutar på rad **36**. Nästlad i den finns en annan punktlista som börjar på rad **11** och slutar på rad **31**. Intressant är nu att den nya, nästlade punktlistan automatiskt väljer en annan än den överordnade punktlistan. `` som börjar på rad **7** får en vanlig punkt (bullet), även kallad `disc`, som listsymbol, medan den nästade punktlistan som börjar på rad **11** får ihålliga ringar, kallade `circles`, se körresultat på nästa sida.

Nästa nästling – som inte finns med i scriptet – skulle få rutor, kallade *squares*. Det är även möjligt att ange ett `type`-attribut till punktlistor genom att använda dessa namn: "`disc`", "`circle`", eller "`square`".

De andra nästlingarna kan spåras i körresultatet av scriptet `List` som följer. Studera noga alla andra nästlingar genom att jämföra körresultatet på nästa sida med koden på förra sidan.

God programmeringsstil

Av skäl som beträffar *God programmeringsstil* måste koden återspegla nätlingarnas logik och struktur genom att göra exakt de kodindragningar du ser i scriptet `List` (sid 38). Utan dessa indragningar blir koden värdelös ur de kriteriernas synpunkt som definierar God programmeringsstil, nämligen:

- Läslighet
- Förståelighet
- Ändringsbarhet

Nedan följer scriptet `Lists` körresultat:



Nästlade och ordnade listor

localhost:51148/List.html

Internets bästa egenskaper:

- Du kan träffa folk från hela världen
- Du får tillgång till nya media så snart de publiceras:
 - Nya spel
 - Nya applikationer:
 - I. för jobb
 - II. för nöje
 - Aktuella nyheter
 - Sökmaskiner
 - Shopping
 - Programmering:
 - a. HTML
 - b. Java
 - c. Python
- Länkar
- Hålla kontakt med gamla kompisar
- Rapportera revolutionära händelser från diktatoriska länder

Mina tre favorit språk:

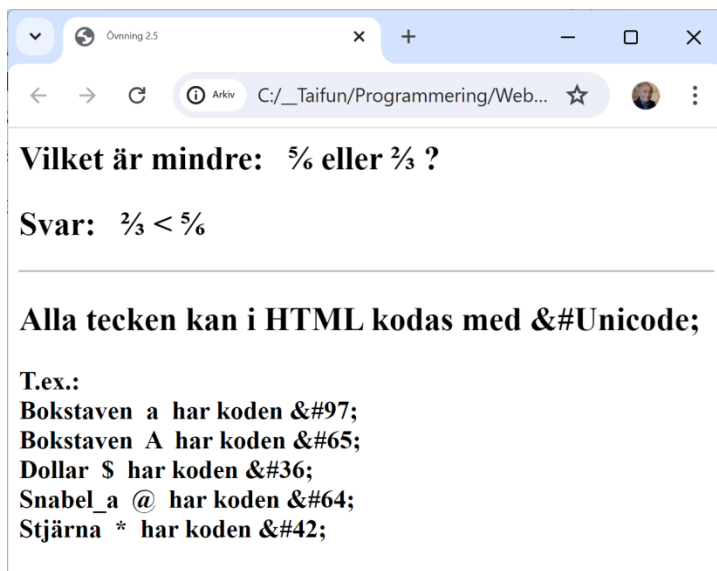
1. HTML
2. CSS
3. JavaScript

- 2.1 Med vilket HTML-element kan man skapa länkar i sitt dokument?
- 2.2 Vad åstadkommer elementtyperna `p`, `b` och `a` i HTML?
- 2.3 Är ankaret ett tomt HTML-element?
- 2.4 Vad är ett *attribut* och var någonstans placeras det i HTML-kod?
- 2.5 Vad gör attributet `href` i ett ankare?
- 2.6 Vilket är det viktigaste attributet av HTML-elementet *ankare*?
- 2.7 Ge tre exempel på attributvärde till attributnamnet `href`.
- 2.8 Hur inleds ett attributvärde till en webbadress?
- 2.9 Hur inleds ett attributvärde till en mailadress?
- 2.10 Vad är förklaringen till beteckningen *ankare* för HTML-elementet `a`?
- 2.11 Vilken egenskap får texten i dokumentet som skrivs i ett ankare?
- 2.12 Kan man även ange namnet på en bildfil som innehåll i ett ankare?
- 2.13 Vilka bildformat är de mest vanliga på webben?
- 2.14 Vilka är de vanligaste bildformat i ett HTML-script?
- 2.15 Till vilket HTML-element hör attributen `height` och `width`?
- 2.16 Vad gör `img`-elementet i HTML?
- 2.17 Vilket är det viktigaste attributet av HTML-elementet `img`?
- 2.18 Hur bestämmer man värdena till attributen `height` och `width`?
- 2.19 Vad gör attributet `alt` i `img`-elementet?
- 2.20 Var någonstans måste du lagra dina bilder när du vill hämta dem med `img`?
- 2.21 Vilka är de mest använda attribut till `img`-elementet?
- 2.22 Hur anger man bildstorleken i `img`-elementet?
- 2.23 Koda några exempel på `img`-elementet.
- 2.24 På vilken mapp pekar `img`-elementets attribut `src` primärt?

- 2.25 Vad exakt händer när man nästlar ett **img**-element i ett ankare?
- 2.26 Är **img**-elementet tomt? Om ja, varför? Om nej, vad är dess innehåll?
- 2.27 Ange två varianter att avsluta **img**-elementet.
- 2.28 Vad händer om man inte specificerar bildstorleken i **img**-elementet?
- 2.29 Formulera med egna ord regeln för att avsluta tomma element.
- 2.30 Ange några exempel på tomma element.
- 2.31 När man i navigeringsmenyn producerad av scriptet **Nav** (sid 30) klickar på bilden som har texten **Tables Page**, visas ett dokument som tidigare skapats av scriptet **Break** (sid 18). Förklara varför?
- 2.32 Ange några specialtecken som är reserverade för att skriva HTML-kod.
- 2.33 Ange den generella syntaxen för kodning av specialtecken i HTML.
- 2.34 Ange några specialtecken som introduceras i scriptet **Contact2** (sid 33).
- 2.35 Kan specialtecken även kodas med sina resp. teckenkoder istället för namn?
- 2.36 Vilken teckenstandard används för kodning av specialtecken med koder?
- 2.37 Ange ett exempel på kodning av specialtecken med teckenkoder.
- 2.38 Vad är *ASCII* och i vilken relation står den till *Unicode*?
- 2.39 Hur ritas man en rak linje i ett HTML-dokument?
- 2.40 Varför kan man koda **hr**-taggen på två olika sätt?
- 2.41 Kodas en oordnad punktlista med **ul** eller **ol**?
- 2.42 Hur ställer upp **ul** sina items?
- 2.43 Genererar **ul** sina items automatiskt?
- 2.44 Med vilken elementtyp kodar man *items* i en punktlista?
- 2.45 Vad gör **li**-taggen?
- 2.46 Vad är skillnaden (i kod och i utskrift) mellan scripten **Links** (sid 23) och **Links2** (sid 36)?
- 2.47 Vilket element har **li**-elementet som innehåll i scriptet **Links2** (sid 36)?
- 2.48 Var någonstans finns i scriptet **Links2** nästlade element?

- 2.49 Vilka listsymboler finns i HTML till förfogande för punktlistor?
- 2.50 Vad är skillnaden mellan elementtyperna **ul** och **ol**?
- 2.51 Vad händer med listsymbolen i punktlistor när man nästlar dem?
- 2.52 Hur bör den hierarkiska strukturen i nästlade listor understrykas i kod?
- 2.53 Kan man själv bestämma listsymbolen i listor?
- 2.54 Vad gör attributet **type** i listor? Kan det användas i både **ul** och **ol**?
- 2.55 Vad händer om man i ordnade listor om man utelämnar attributet **type**?
- 2.56 Med vilket värde till **type** kan man få stora bokstäver som listsymboler?

- 2.1 Kör scriptet **Links** (sid 24) i din webbläsare. Ladda scriptet i din favorit editor. Avlägsna alla paragraftaggar och kör igen. Ser du någon skillnad i körresultatet? Svaret är beroende på vilken webbläsare du använder. Försök att genomföra experimentet i olika webbläsare. Vilken slutsats drar du?
- 2.2 Scriptet **Contact** (sid 25) har på rad **14** ett radbyte i koden. Medför detta även radbyte i dokumentet? Testa! Lägg in flera tomma rader i koden och testa igen. Lägg in slutligen HTML-kod för radbyte på samma ställe och kör. Testa de olika varianterna av break-taggen.
- 2.3 Scriptet **Picture** (sid 27) kommer inte att visa några bilder i din webbläsare, så länge bildfilerna ***.jpg** som är angivna som värden till attributet **src** inte finns på din dator och dessutom inte på rätt plats. Vad ser du istället? Ersätt bildfilerna med egna bilder och placera dem på rätt plats. Justera bildernas storlek. Modifiera attributet **alt** meningsfullt och testa.
- 2.4 För scriptet **Nav** (sid 30) gäller samma sak som sades i övn 2.3 ovan om bilder. Det tillkommer här även att scriptfilerna ***.html** inte heller finns på de angivna platserna hos dig. Modifiera scriptet **Nav** med egna bilder och scriptfiler på korrekta platser, så att allt fungerar på ett meningsfullt sätt.
- 2.5 Modifiera scriptet **Contact2** (sid 33) så att det producerar nedanstående körresultat. Dvs använd *talkoder* istället för *namn* för att koda tecken i HTML. Förfarandet beskrivs under *Talkoder* på sid 34.



- 2.6 Vidareutveckla scriptet **Links2** (sid 36) så att det producerar de fem sista raderna i utskriften av övn. 2.5 (ovan) som en ordnad punktlista. Fortsätt att använda **Talkoder** istället för *namnkoder* för att koda specialtecken i HTML enligt förfarandet som beskrivs på sid 34.
- 2.7 Skriv ett script **MyList** som liknar scriptet **List** (sid 38) och med hjälp av ordnade, oordnade och nästlade listor genererar en utskrift liknande den på sid 40. Ditt körresultat ska ge svar på frågan vilka som är Internets *sämsta* egenskaper samt vilka dina tre favoritämnen i skolan är.

Beakta att din kod måste uppfylla alla kriterier på *God programmeringsstil* (sid 39), speciellt när det gäller indragningarna i koden. Dessa måste överensstämma med strukturen och logiken i de nästlingar du gör med dina listor. Försök att lära dig detta genom att jämföra scriptet **List** med dess körresultat på sid 40.

Kapitel 3

Mer om HTML

Ämne	Sida	Program
3.1 Tabeller	47	Tabell11
- <code><table></code> -attributet <code>border</code>	47	
- Elementet <code>table</code>	48	
- Elementen <code>th</code> och <code>td</code>	48	
3.2 En mer utvecklad tabell	49	Tabell12
- <code>th</code> - och <code>td</code> -attributen <code>rowspan</code> och <code>colspan</code>	50	
3.3 HTML Forms	51	Form1
- Elementet <code>form</code>	51	
- JavaScript funktion	52	
- Elementet <code>input</code>	52	
3.4 En mer utvecklad Form	53	Form2
- Elementet <code>textarea</code>	54	
- Maskerad textbox	54	
- Checkboxar	54	
3.5 Radioknappar och Dropp-down list	55	Form3
- Radioknappar	55	
- Dropp-down list	57	
3.6 Interna länkar	58	Intern_Links
- Namngivna ankare	58	
3.7 Interna länkar i andra dokument	61	Extern_Link
- Sökväg som referens	61	Intern_Links_2
3.8 Image maps	64	Picture
- Elementet <code>map</code>	64	
- Elementet <code>area</code>	65	
- Koordinatsystem för geometriska figurer	65	
Övningar till kap 3	68	

3.1 Tabeller

```
1 <!-- Tabell1.html -->
2 <head>
3   <title>En enkel HTML-tabell</title>
4 </head>
5 <body>
6   <table border = "1" width = "50%">           <!-- Skapar tabell (6-28) -->
7     <caption><b>Pris för Frukt</b></caption>    <!-- Tabellens rubrik -->
8     <thead>                                   <!-- Tabellhuvudets område (8-13) -->
9       <tr>                                     <!-- tr = table row infogar en rad -->
10        <th>Frukt</th>                         <!-- th infogar en huvudcell (fet) -->
11        <th>Pris</th>
12      </tr>
13    </thead>
14    <tbody>                                     <!-- Tabellkroppens område (14-22) -->
15      <tr>
16        <td>Äpple</td>                         <!-- td infogar en data cell -->
17        <td>5 kr</td>
18      </tr>
19      <tr> <td>Orange</td> <td>10 kr</td> </tr>
20      <tr> <td>Banan</td> <td>15 kr</td> </tr>
21      <tr> <td>Ananas</td> <td>20 kr</td> </tr>
22    </tbody>
23    <tfoot>                                     <!-- Tabellfotens område (23-27) -->
24      <tr>
25        <th>Total</th> <th>50 kr</th> <!-- 2 huvudceller (fet) -->
26      </tr>
27    </tfoot>
28  </table>
29 </body>
```

table-attributet border

På rad 6 skapar **table**-taggen en tabell som har två attribut, av vilka attributet **border** ritar en ram, dvs kantlinjerna till tabellen. Attributets värde bestämmer ramens typ och stil. Utan att definiera attributet **border** inramas tabellen inte alls, vilket inte ger intrycket av en "riktig" tabell. I scriptet ovan har attributet **border**:s värde

satts till **1** vilket ritar den ram som ses på bilden ovan. Prova gärna andra värden **0**, **2**, **3**, ... för att se andra typer av kantlinjer. Enheten är pixlar. **0** betyder ingen ram. Det andra attributet **width** bestämmer tabellens bredd. I scriptet **Table1** är breddens enhet angiven i procent av webbläsarens bredd. För att få en uppfattning om

Frukt	Pris
Äpple	5 kr
Orange	10 kr
Banan	15 kr
Ananas	20 kr
Total	50 kr

den faktiska storleken, prova att köra scriptet med olika värden för **width** och olika storlekar på webbläsaren.

Elementet **table**

Tabellen ovan initieras i scriptet **Tabell11** på förra sidan av starttaggen **<table>** på rad **6** och avslutas av sluttaggen **</table>** på rad **28**. Som bilden visar har tabellen 6 rader och 2 kolumner och en rubrik. Rubriken skapas med taggen **caption** och hamnar utanför (ovanpå) tabellen. Den första och sista raden är tabellens huvud (*head*) och fot (*foot*) och har annorlunda formateringar: fet stil och centrerad, vilket beror på att deras koder placerats i speciella taggar. Dvs den första raden kodas i tabellhuvudets område **thead** med **th**-celler (rad **10-11**). Och den sista raden kodas i tabellfotens område **tfoot**, även den med **th**-celler (rad **25**). Båda omgärdar tabellens kropp. Ur ett helhetsperspektiv kan vi konstatera:

Den överordnade strukturen i tabellen består av följande tre delar:

- Huvudet (*head*) som definieras med ett **thead**-element (raderna **8-13**)
- Kroppen (*body*) som definieras med ett **tbody**-element (raderna **14-22**)
- Foten (*foot*) som definieras med ett **tfoot**-element (raderna **23-27**).

Elementen **th** och **td**

th står för *table header column* och **th** för *table data*. Båda skapar kolumner.

Varje **tr**-element (t.ex. rad **9-12**) definierar en enskild tabellrad, medan kolumnerna i huvudets område skapas med ett **th**-element som formaterar texten i fet stil och centrerad.

Tabellkroppens område (rad **14-22**) innehåller tabellens primära data och är definierad med ett **tbody**-element. Även där skapas raderna med **tr**-element, medan kolumnerna kodas med **td**. I vanliga dataceller som skapas med **td** sker ingen formatering.

3.2 En mer utvecklad tabell

```
1 <!-- Tabell2.html -->
2 <head>
3   <title>En mer utvecklad tabell</title>
4 </head>
5 <body>
6   <h1>Exempel på en mer utvecklad tabell</h1>
7   <table border = "3"> <!-- Tabell (7-36) -->
8     <caption>Så här ser en mer utvecklad tabell ut:</caption>
9     <thead> <!-- Tabellhuvud (9-23) -->
10      <tr>
11        <th rowspan = "2"> <!-- Slår ihop 2 rader -->
12          <img src = "camel.gif" width = "210" height = "170"
13            alt = "Kamelbild">
14        </th>
15        <th colspan = "4" valign = "top"> <!-- låår ihop 4 kolumner -->
16          <h1>Kameldjur jämförelse</h1> <br> Uppdaterat 10/2024
17        </th>
18      </tr>
19      <tr valign = "top"> <!-- Vertikal justering -->
20        <th># Pucklar</th> <th>Ursprungsregion</th> <th>Spotta?</th>
21        <th>Producerar ull?</th>
22      </tr>
23    </thead>
24    <tbody> <!-- Tabellkropp (24-35) -->
25      <tr>
26        <th>Kameler (baktriska)</th> <td>2</td> <td>Afrika/Asien</td>
27        <td rowspan = "2">Lama</td>
28        <td rowspan = "2">Lama</td> <!-- Slår ihop 2 rader -->
29      </tr>
30      <tr>
31        <th>Lamor</th>
32        <td>1</td>
33        <td>Anderna (Sydam.)</td>
34      </tr>
35    </tbody>
36  </table>
37 </body>
```

På rad 7 skapar `table`-taggen en tabell med värdet "3" till attributet `border` lite annorlunda kantlinjer än förra tabellen, se körresultatet till höger.

Exempel på en mer utvecklad tabell

Så här ser en mer utvecklad tabell ut:

	Kameldjur jämförelse			
	Uppdaterat 10/2024			
	# Pucklar	Ursprungsregion	Spotta?	Producerar ull?
Kameler (baktriska)	2	Afrika/Asien	Lama	Lama
Lamor	1	Anderna (Sydam.)		

th- och td-attributen rowspan & colspan

Tabellceller formateras så att de kan visa data som de innehåller. Men utvecklare kan skapa större dataceller med attribut som de skriver till cell-taggarne **td** och **th**. Dessa attribut heter **rowspan** och **colspan**. Värden de får, bestämmer antalet rader resp. kolumner som reserveras för en cell. Två exempel har vi för detta i scriptet **Tabe112** på förra sidan:

- **th**-elementet på raderna **11-14** använder attributet **rowspan = "2"** för att tillåta cellen, som innehåller kamelbilden, att använda 2 vertikala grann-celler. Dvs det slår ihop (*spans*) 2 rader.
- **th**-elementet på raderna **15-17** använder attributet **colspan = "4"** för att tillåta cellen, som innehåller texten **Kameldjur jämförelse** och **Uppdaterad 10/2024**, att använda 4 horisontella grann-celler, dvs att slå ihop (*span*) 4 kolumner.

Motsvarande gäller för **td**-elementen på raderna **27-28**.

Oftast kommer en sådan form från en server på nätet och går också tillbaka till den, efter att användaren har fyllt i den. Vi kan inte genomföra den här kommunikationen utan bara simulera den, för att lära oss hur man kodar former i HTML. Istället för att skicka formuläret till en server skriver vi ut ett svar med hjälp av en JavaScript funktion.

JavaScript funktion

I scriptet **Form1**, raderna **24-30**, har vi definierat funktionen `showData()` som är kodad i *JavaScript* och genererar en meddelanderuta som kallas *alert box*, se förra sidan. JavaScript är ett scriptspråk som kan bäddas in i HTML och har element av de universella programmeringsspråken som saknas i HTML. `<script>`-taggen på rad **24** inleder JavaScript-kod och sluttaggen på rad **30** avslutar den. Kommentarer i JavaScript kodas med `//` (raderna **24 & 25**). Närmare bestämt inleds en *radkommentar* med `//` som inte behöver avslutas då den gäller till slutet av raden.

På rad **18** anropas funktionen `showData()` i `input`-attributet `onClick`, närmare bestämt i attributets värde, vilket genererar alert-boxen som man ser på förra sidan (nedan till höger). Anropet sker när användaren klickar på knappen Skicka din respons i formuläret. Detaljerna i definitionen till JavaScript-funktionen `showData()` kommer att tas upp senare resp. i kursen Webbutveckling 2.

Textboxar med elementet input

Det är elementet `input` med sina attribut som i huvudsak designar formen, genom att skapa textboxar och knappar på formen. Det är attributet `type` som bestämmer vilken typ av `input` det ska bli. Medan `type = "text"` (rad **11**) infogar en textbox, skapar `type = "submit"` (rad **17**) en knapp som skickar formen och `type = "reset"` (rad **20**) en annan knapp som tömmer allt.

3.4 En mer utvecklad Form

I förra avsnitt introducerades grundläggande HTML Forms med enkla element som **form**, **input** samt deras olika attribut. I detta avsnitt vill vi fortsätta med lite mer avancerade element. Scriptet **Form2** innehåller elementet **textarea** och nya varianter av textboxar av typ password (maskerad textbox) samt checkbox:

```
Solution1
Form2.html
1 <!-- Form2.html -->
2 <title>En mer utvecklad Form</title>
3 <h1>Feedback Form</h1>
4 <p>Fyll i detta formulär för att hjälpa oss att förbättra hemsidan.</p>
5 <form> <!-- Form skapas -->
6 <p>
7 <label>
8   Namn: <!-- Textbox skapas för namn -->
9   <input type="text" id="Name" size="25">
10 </label>
11 </p>
12 <p>
13 <label>
14   Kommentar:<br> <!-- Multiline Textbox skapas för kommentar -->
15   <textarea name="comments" rows="4" cols="36">
16     >Skriv här dina kommentarer.</textarea>
17 </label>
18 </p>
19 <p>
20 <label>
21   E-mail adress: <!-- Textbox skapas med maskerad text -->
22   <input type="password" name="email" size="25">
23 </label>
24 </p>
25 <p>
26 <b>Vad gillar du på hemsidan?</b><br>
27 <label>
28   Designen <!-- 5 checkboxar skapas: -->
29   <input name="thingsliked" type="checkbox" value="Design">
30 </label>
31 <label>
32   Länkarna
33   <input name="thingsliked" type="checkbox" value="Links">
34 </label>
35 <label>
36   Användarvänligheten
37   <input name="thingsliked" type="checkbox" value="Ease">
38 </label>
39 <label>
40   Bilderna
41   <input name="thingsliked" type="checkbox" value="Images">
100% No issues found Ln: 8 Ch: 58 SPC CRLF
```

... (Koden fortsätter på nästa sida)

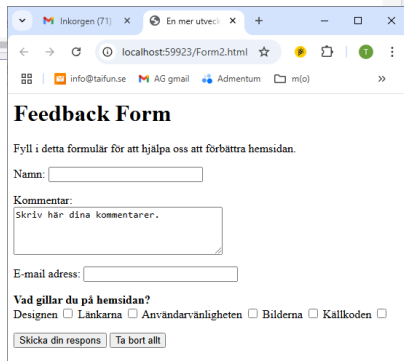
```

Solution1
Form2.html
42     </label>
43     <label>
44         Källkoden
45         <input name="thingsliked" type="checkbox" value="Code">
46     </label>
47 </p>
48 <p>                                     <!-- Anrop av JS fkt.: -->
49     <input type="submit" value="Skicka din respons" onClick="showData()">
50     <input type="reset" value="Ta bort allt">
51 </p>
52 </form>
53
54 <script>                               // JavaScript funktion
55     function showData()
56     {
57         user = document.getElementById("Name").value;           // Hämtar texten
58         alert("Hej " + user + "! " + "\nTack för din respons.") // från textboxen
59     }
60 </script>

```

Elementet textarea

På raderna 15-16 skapas en *multiline textbox* med HTML-elementet **textarea**, där man till skillnad från en vanlig **input**-textbox, kan skriva flera rader. Exakt hur många, kan man bestämma själv genom att ange antalet i attributen **rows** och **cols** (rad 15). En s.k. *default text* som ska redan finnas i textarean, kan man placera i koden som innehåll till elementet **textarea**, dvs mellan start- och sluttaggen (rad 16).



Maskerad textbox

På rad 22 skapas med elementet **input** en – till att börja med – vanlig textbox. Men genom att ge dess attribut **type** värdet "password" blir texten som man sedan skriver i den i dokumentet, maskerad. Det innebär att alla bokstäver man matar in, ersätts i dokumentet med asterisk (*), medan textboxen sparar den inmatade texten i sitt variabelnamn (i **id**-attributets värde). Det är så man vill ha det i en textbox av typ **password**. Här har man valt att inte offentligt visa mailadressen.

Checkboxar

En checkbox är en liten ruta (☐) som kan bockas. På rad 29 skapas den första checkboxen i scriptet **Form2** (sid 53). För det används en vanlig textbox. Men genom attributet **type = "checkbox"** blir denna textbox en *checkbox*. I ett formulär kan man bocka *flera* checkboxar.

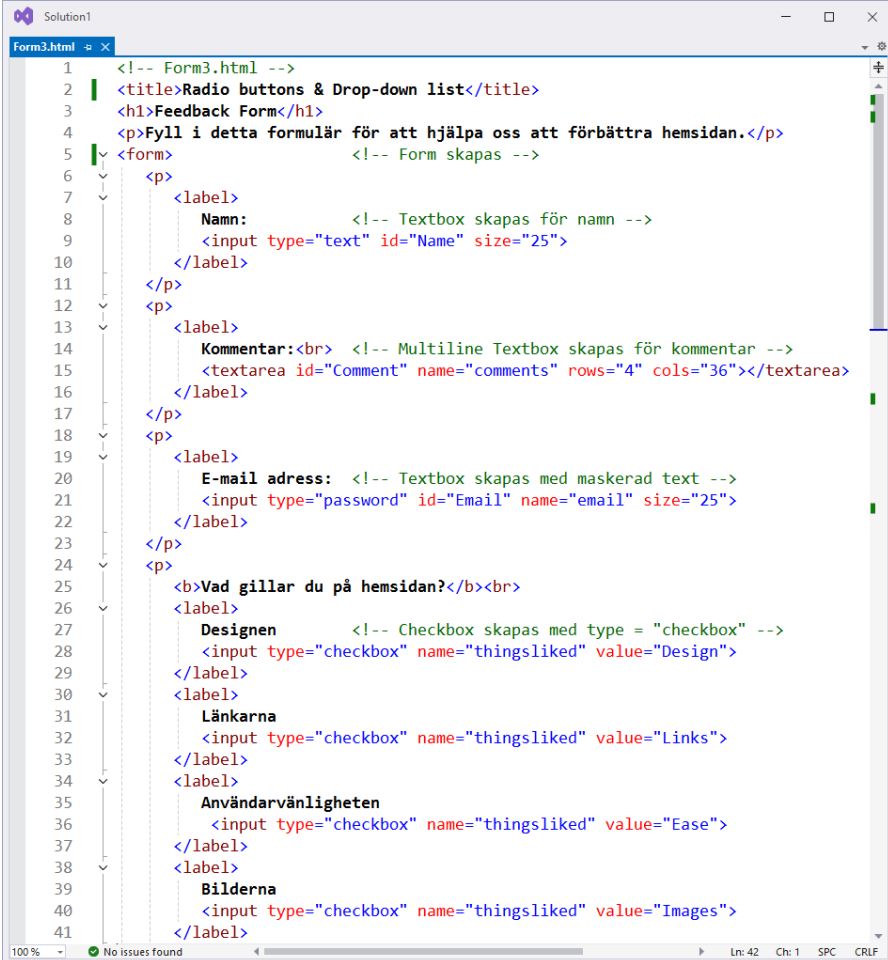
På raderna 33, 37, 41 och 45 skapas ytterligare 4 checkboxar. Deras namn bestäms i elementet **label** som omgärdar checkboxarna. I dokumentet (ovan) visas de under rubriken *Vad gillar du på hemsidan?* i nästsista raden.

3.5 Radioknappar och Drop-down list

Den nya, utvidgade formen **Form3** nedan tar över **Form2**, modifierar den lite och lägger till två nya element: **radioknappar** och en **drop-down list**.

Radioknappar

En radioknapp ser ut som en ring (○). Till skillnad från checkboxar kan man i ett formulär markera endast en radioknapp (*radio button*).



```
1 <!-- Form3.html -->
2 <title>Radio buttons & Drop-down list</title>
3 <h1>Feedback Form</h1>
4 <p>Fyll i detta formulär för att hjälpa oss att förbättra hemsidan.</p>
5 <form> <!-- Form skapas -->
6 <p>
7 <label>
8     Namn: <!-- Textbox skapas för namn -->
9     <input type="text" id="Name" size="25">
10 </label>
11 </p>
12 <p>
13 <label>
14     Kommentar:<br> <!-- Multiline Textbox skapas för kommentar -->
15     <textarea id="Comment" name="comments" rows="4" cols="36"></textarea>
16 </label>
17 </p>
18 <p>
19 <label>
20     E-mail adress: <!-- Textbox skapas med maskerad text -->
21     <input type="password" id="Email" name="email" size="25">
22 </label>
23 </p>
24 <p>
25     <b>Vad gillar du på hemsidan?</b><br>
26     <label>
27         Designen <!-- Checkbox skapas med type = "checkbox" -->
28         <input type="checkbox" name="thingsliked" value="Design">
29     </label>
30     <label>
31         Länkarna
32         <input type="checkbox" name="thingsliked" value="Links">
33     </label>
34     <label>
35         Användarvänligheten
36         <input type="checkbox" name="thingsliked" value="Ease">
37     </label>
38     <label>
39         Bilderna
40         <input type="checkbox" name="thingsliked" value="Images">
41     </label>
```

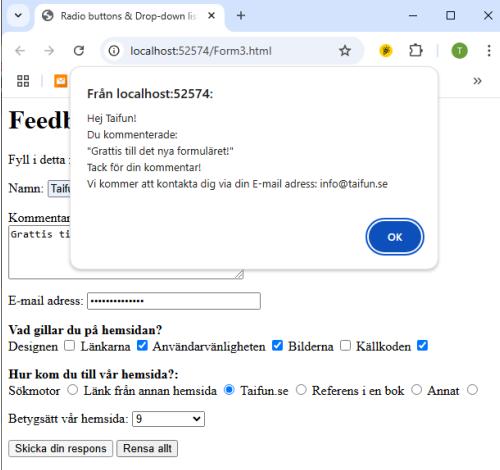
... (Koden fortsätter på nästa sida)

```
Solution1
Form3.html
42 <label>
43     Källkoden
44     <input type="checkbox" name="thingsliked" value="Code">
45 </label>
46 </p>
47 <p>
48     <b>Hur kom du till vår hemsida?:</b><br>
49 <label>
50     Sökmotor           <!-- Radio button skapas med type = "radio" -->
51     <input name="howtosite" type="radio" value="search engine" checked="checked">
52 </label>
53 <label>
54     Länk från annan hemsida
55     <input name="howtosite" type="radio" value="link">
56 </label>
57 <label>
58     Taifun.se
59     <input name="howtosite" type="radio" value="taifun.se">
60 </label>
61 <label>
62     Referens i en bok
63     <input name="howtosite" type="radio" value="book">
64 </label>
65 <label>
66     Annat
67     <input name="howtosite" type="radio" value="other">
68 </label>
69 </p>
70 <p>
71 <label>
72     Betygsätt vår hemsida:
73     <select name="rating">           <!-- Dropp-down list skapas -->
74         <option>Fantastiskt</option>   <!-- 12 alternativ -->
75         <option>10</option>
76         <option>9</option>
77         <option>8</option>
78         <option>7</option>
79         <option>6</option>
80         <option>5</option>
81         <option>4</option>
82         <option>3</option>
```

```
Solution1 - form3.html
form3.html
83         <option>2</option>
84         <option>1</option>
85         <option>Awful</option>
86     </select>
87 </label>
88 </p>
89 <p>
90     <input type="submit" value="Skicka din respons" onClick="showData()">
91     <input type="reset" value="Rensa allt">
92 </p>
93 </form>
94
95 <script>
96     function showData()
97     {
98         user = document.getElementById("Name").value           // Hämtar texten
99         comment = document.getElementById("Comment").value     // från textboxarna
100        email = document.getElementById("Email").value         // Name, rad 9
101        alert("Hej " + user + "!" + "\nDu kommenterade:\n" +   // Comment, rad 15
102              "\"\" + comment + "\"\nTack för din kommentar!\n" + // och Email, rad 21
103              "Vi kommer att kontakta dig via din E-mail adress: " + email)
104    }
105 </script>
```


Drop-down list

I scriptet **Form3** (ovan) skapas en *drop-down list* med HTML-elementet **select** (raderna 73-86). I dokumentet (till höger) ser man den på nästsista raden höger om texten Betygsätt vår hemsida: . Det är en rullgardinsmeny som låter användaren, efter man har klickat på den lilla pilen, välja mellan ett antal alternativ som i koden skapas med **option**-taggen inbäddad i **select** (raderna 74-85). På så sätt ges användaren möjligheten att betygsätta hemsidan med 12 olika alternativ.



The screenshot shows a web browser window with the address bar displaying 'localhost:52574/Form3.html'. An alert box is open, titled 'Från localhost:52574:', with the following text: 'Hej Taifun! Du kommenterade: "Grattis till det nya formuläret!" Tack för din kommentar! Vi kommer att kontakta dig via din E-mail adress: info@taifun.se'. Below the alert box, the form is visible. It includes a 'Feedb' section with a 'Fyll i detta:' label, a 'Namn:' field containing 'Taifu', and a 'Kommentar' field containing 'Grattis ti'. Below these is an 'E-mail adress:' field. The form also has a section 'Vad gillar du på hemsidan?' with checkboxes for 'Designen', 'Länkarna', 'Användarvänligheten', 'Bilderna', and 'Källkoden'. The 'Användarvänligheten' and 'Källkoden' checkboxes are checked. Below this is a section 'Hur kom du till vår hemsida?:' with radio buttons for 'Sökmotor', 'Länk från annan hemsida', 'Taifun.se', 'Referens i en bok', and 'Annat'. The 'Taifun.se' radio button is selected. Below this is a 'Betygsätt vår hemsida:' dropdown menu with the value '9' selected. At the bottom of the form are two buttons: 'Skicka din respons' and 'Rensa allt'.

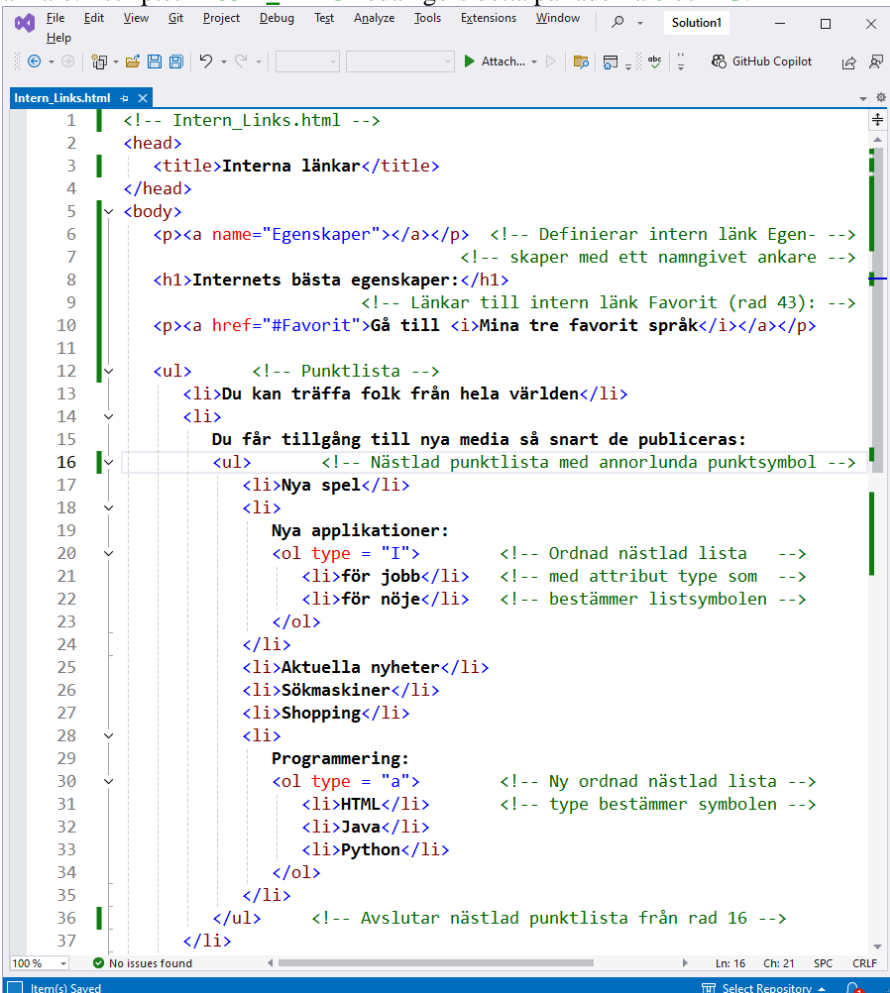
Dessutom har implementerar scriptet **Form3** övn 3.5:s (sid 69) krav på att hämta data inte bara från textboxen Namn (rad 9), utan även från textarean Kommentar (rad 15) och från textboxen E-mail adress (rad 22) samt visa dessa data i JavaScript funktionens alert-box, se utskriften ovan. Koden för dessa tillägg finns på de angivna raderna.

3.6 Interna länkar

I detta avsnitt tar vi scriptet **List** (sid 38) som har körresultatet på sid 40 och modifierar den genom att lägga till två *interna länkar*. Vanliga länkar hade vi lärt känna i scriptet **Links** (sid 24) som vi kodade med HTML-elementet ankare **a** (sid 25). Medan vanliga länkar går till andra dokument, hoppar interna länkar mellan olika ställen i *samma* dokument, vilket är av intresse i stora dokument. Interna länkar skapas med ankare som har ett namn, s.k.:

Namngivna ankare

Genom att definiera ett värde till attributet **name** i ett ankare, skapas ett namngivet ankare. I scriptet **Intern_Links** nedan görs detta på raderna **6** och **43**.



```
1 <!-- Intern_Links.html -->
2 <head>
3   <title>Interna länkar</title>
4 </head>
5 <body>
6   <p><a name="Egenskaper"></a></p> <!-- Definierar intern länk Egen- -->
7     <!-- skaper med ett namngivet ankare -->
8   <h1>Internets bästa egenskaper:</h1>
9     <!-- Länkar till intern länk Favorit (rad 43): -->
10  <p><a href="#Favorit">Gå till <i>Mina tre favorit språk</i></a></p>
11
12  <ul> <!-- Punktlista -->
13    <li>Du kan träffa folk från hela världen</li>
14    <li>
15      Du får tillgång till nya media så snart de publiceras:
16      <ul> <!-- Nästlad punktlista med annorlunda punktsymbol -->
17        <li>Nya spel</li>
18        <li>
19          Nya applikationer:
20          <ol type = "I"> <!-- Ordnad nästlad lista -->
21            <li>för jobb</li> <!-- med attribut type som -->
22            <li>för nöje</li> <!-- bestämmer listsymbolen -->
23          </ol>
24        </li>
25        <li>Aktuella nyheter</li>
26        <li>Sökmaskiner</li>
27        <li>Shopping</li>
28        <li>
29          Programmering:
30          <ol type = "a"> <!-- Ny ordnad nästlad lista -->
31            <li>HTML</li> <!-- type bestämmer symbolen -->
32            <li>Java</li>
33            <li>Python</li>
34          </ol>
35        </li>
36      </ul> <!-- Avslutar nästlad punktlista från rad 16 -->
37    </li>
38  </ul>
```

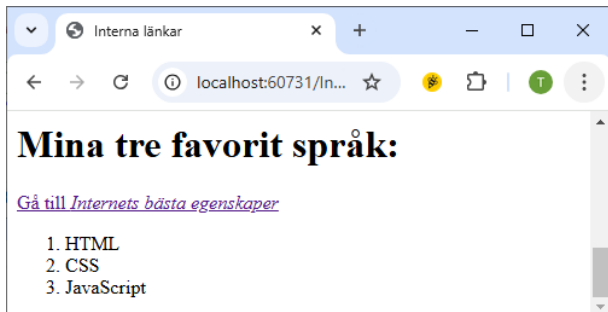
... (Koden fortsätter på nästa sida)

```
File Edit View Git Project Debug Test Analyze Tools Extensions Window Search - Solution1
Help
Intern_Links.html
38 <li>Länkar</li>
39 <li>Hålla kontakt med gamla kompisar</li>
40 <li>Rapportera revolutionära händelser från diktatoriska länder</li>
41 </ul> <!-- Avslutar punktilista från rad 12 -->
42
43 <p><a name="Favorit"></a></p> <!-- Definierar intern länk -->
44 <!-- Favorit med ett namngivet ankare -->
45 <h1>Mina tre favorit språk:</h1>
46 <!-- Länkar till intern länk Egenskaper (rad 6): -->
47 <p><a href="#Egenskaper">Gå till <i>Internets bästa egenskaper</i></a></p>
48
49 <ol> <!-- Ordnad lista utan attributet type -->
50 <li>HTML</li> <!-- får en numerisk sekvens 1, 2, 3, ... -->
51 <li>CSS</li>
52 <li>JavaScript</li>
53 </ol>
54 </body>
```

Kör man scriptet **In-tern_Links** ovan, måste man minska utskriftsfönstret *innan* man klickar på länkarna, för att se bytet av platsen i dokumentet.

Den interna länken som skapas med ankare på rad **10**, leder till den del av dokumentet som är rubricerad med **Mina tre favorit språk**. Medan länken som skapas med ankare på rad **47**, leder till den del av dokumentet som är rubricerad med **Internets bästa egenskaper**.

Observera att varje intern länk har *två* två olika platser i koden: en gång definitionen av den plats i dokumentet som länken ska leda *till* med ett ankare och attri-



butet `name` (raderna **6** och **43**), en annan gång referensen som man ska klicka på med ett ankare och attributet `href` (raderna **10** och **47**). Det är som i all kommunikation: det finns en sändare och mottagare. Först måste mottagarens plats definieras (`name`), sedan kan sändaren länka till mottagaren (`href`).

3.7 Interna länkar i andra dokument

Scriptet **Intern_Links** i förra avsnitt (sid 58) definierade interna länkar som hoppade mellan olika ställen i *samma* dokument.

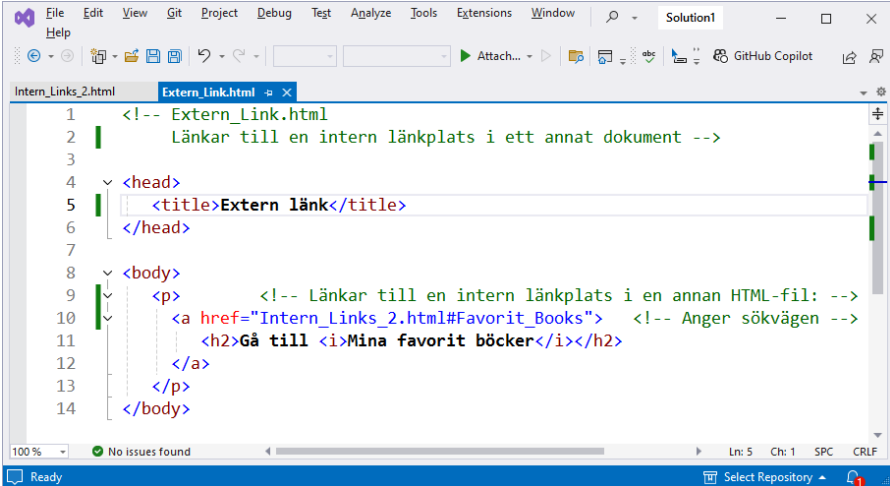
Här ska vi skapa länkar som går till interna länkar i andra dokument. Scriptet **Extern_Link** nedan skapar ett ankare vars attribut **href** refererar till en länkplats som är definierad i ett annat dokument. Länkplatsen heter **Favorit_Books** och är definierad i filen **Intern_Links_2.html**.

Sökväg som referens

På rad **10** nedan skapas ett ankare med attributet **href** vars värde är sökvägen till länkplatsen **Favorit_Books**. Observera sökvägens syntax:

Intern_Links_2.html#Favorit_Books

Filen **Intern_Links_2.html** måste placeras i samma mapp som scriptet **Extern_Link** nedan. Väljer man en annan plats måste motsvarande sökväg anges.



```
1 <!-- Extern_Link.html
2 |     Länkar till en intern länkplats i ett annat dokument -->
3
4 <head>
5 |     <title>Extern länk</title>
6 | </head>
7
8 <body>
9 |     <p>
10 |         <a href="Intern_Links_2.html#Favorit_Books"> <!-- Anger sökvägen -->
11 |             <h2>Gå till <i>Mina favorit böcker</i></h2>
12 |         </a>
13 |     </p>
14 </body>
```

Scriptet **Intern_Links_2** nedan som scriptet **Extern_Link** hänvisar till är en modifikation av scriptet **Intern_Links** i förra avsnitt (sid 58). Den första delen med rubriken **Internets bästa egenskaper** är oförändrad. I den andra delen har vi ersatt **Mina favorit språk** med **Mina favorit böcker** och döpt om namnet på ankaret från **Favorit** till **Favorit_Books**.

```
1 <!-- Intern_Links_2.html -->
2 <head>
3   <title>Interna länkar i andra dokument</title>
4 </head>
5 <body>
6   <p><a name="Egenskaper"></a></p>   <!-- Definierar länkplats -->
7                                     <!-- med namnet Egenskaper -->
8   <h1>Internets bästa egenskaper:</h1>
9     <!-- Länkar till intern länkplats Favorit_Books (rad 47): -->
10  <p><a href="#Favorit_Books">Gå till <i>Mina favorit böcker</i></a></p>
11
12 <ul>
13   <!-- Punktlista -->
14   <li>Du kan träffa folk från hela världen</li>
15   <li>
16     Du får tillgång till nya media så snart de publiceras:
17     <ul>
18       <!-- Nästlad punktlista med annorlunda punktsymbol -->
19       <li>Nya spel</li>
20       <li>
21         Nya applikationer:
22         <ol type="I">
23           <!-- Ordnad nästlad lista -->
24           <li>för jobb</li>   <!-- med attribut type som -->
25           <li>för nöje</li>   <!-- bestämmer listsymbolen -->
26         </ol>
27       </li>
28       <li>Aktuella nyheter</li>
29       <li>Sökmaskiner</li>
30       <li>Shopping</li>
31       <li>
32         Programmering:
33         <ol type="a">
34           <!-- Ny ordnad nästlad lista -->
35           <li>HTML</li>   <!-- type bestämmer symbolen -->
36           <li>Java</li>
37           <li>Python</li>
38         </ol>

```

... (Koden fortsätter på nästa sida)

```
File Edit View Git Project Debug Test Analyze Tools Extensions Window Help Solution1
Intern Links 2.html
39         </li>
40     </ul>     <!-- Avslutar nästlad punktlista från rad 16 -->
41 </li>
42 <li>Länkar</li>
43 <li>Håll kontakt med gamla kompisar</li>
44 <li>Rapportera revolutionära händelser från diktatoriska länder</li>
45 </ul>     <!-- Avslutar punktlista från rad 12 -->
46
47 <p><a name="Favorit_Books"></a></p>     <!-- Definierar länkplatsen -->
48                                     <!-- Favorit_Books -->
49
50 <h1>Mina favorit böcker:</h1>
51     <!-- Länkar till intern länkplats Egenskaper (rad 6): -->
52 <p><a href="#Egenskaper">Gå till <i>Internets bästa egenskaper</i></a></p>
53 <p>
54     
56     
58 </p>
</body>
```

100% No issues found Ln: 50 Ch: 22 SPC CRLF

Ready Select Repository

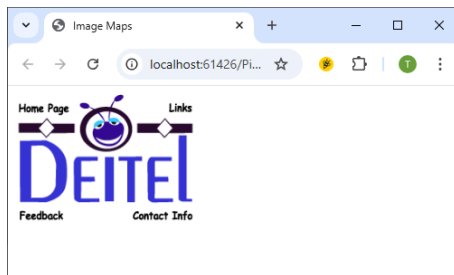
3.8 Image maps

Image maps, på svenska bildkartor, är ett verktyg i HTML som skapar klickbara ytor på en bild. Ytorna är osynliga och kallas för *hotspots*. En image map läggs som en osynlig karta på en bild eller *"bilden använder en image map"*. Så, själva bilden måste skiljas från image mapen. I scriptet **Picture** nedan använder bilden som skapas med **img** på raderna 6-7, image mapen som med **map** skapas på raderna 8-27.

```
MyConsoleProject - Microsoft Visual Studio
extensions  windows  netcp
Debug  x64  Local Windows Debugger  GitHub Copilot
Picture.html  x
1  <!-- Picture.html -->
2  <head>
3      <title>Image Maps</title>
4  </head>
5  <body>                                <!-- Bild som använder image mapen picture: -->
6      <img src = "deitel.gif" width = "200" height = "144"
7          alt = "Deitel logo" usemap = "#picture">
8      <map name = "picture">             <!-- Skapar image mapen picture -->
9          <area href = "Form1.html" shape = "rect"
10             coords = "2, 123, 54, 143"
11             alt = "Go to the feedback form"> <!-- Rektangulär yta på -->
12                 <!-- koordinaterna: (2,123): övre vänstra hörnet -->
13                 <!-- (54,143): nedre högra hörnet -->
14             <area href = "contact.html" shape = "rect"
15                 coords = "126, 122, 198, 143"
16                 alt = "Go to the contact page">
17             <area href = "main.html" shape = "rect"
18                 coords = "3, 7, 61, 25" alt = "Go to the homepage">
19             <area href = "Intern_Links.html" shape = "rect"
20                 coords = "168,5,197,25" alt = "Go to the links page">
21             <area href = "mailto:info@taifun.se" shape = "poly"
22                 coords = "162,25, 154,39, 158,54, 169,51, 183,39, 161,26"
23                 alt = "E-mail Taifun"> <!-- Polygon yta med 6 hörn -->
24             <area href = "mailto:info@taifun.se"
25                 shape = "circle" coords = "100, 36, 33"
26                 alt = "E-mail Taifun"> <!-- Cirkulär yta med -->
27             </map>                       <!-- medelpunkt och radie -->
28  </body>
```

Elementet **map**

Image maps definieras med HTML-elementet **map**. I scriptet ovan skapas en image map och döps med attributet **name** till **picture** (rad 8). Bilden som skapas på raderna 6-7, använder denna map, genom att referera till den med attributet **usemap** (rad 7). Bilden till höger har sex klickbara ytor (hotspots).



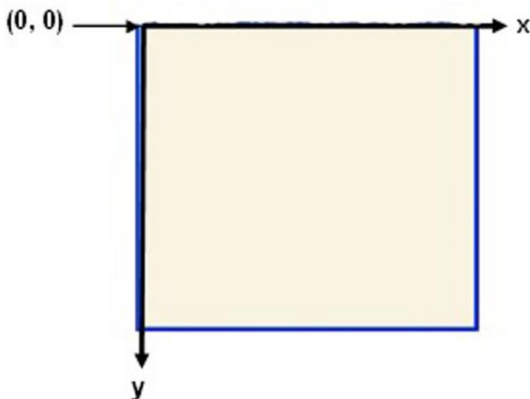
Elementet `area`

De klickbara ytorna i image mapen `picture` skapas med HTML-elementet `area` som nästlas in i `map`-elementet. Så, en `map` kan innehålla en eller flera `area`:s. Ytorna kan ha olika former. Dessa anges med attributet `shape`. I scriptet `Picture` har vi fyra rektanglar, en cirkel och en sexhörning (hexagon). `area`-attributet `href` refererar till de dokument som ska visas, när man klickar på de hotspots i image mapen.

De geometriska figurers position och storlek som är definierade med attributet `shape`, anges i attributet `coords`. För rektangeln anges det övre vänstra och nedre högra hörnets koordinater (raderna `10`, `15`, `18` och `20`). För sexhörningen anges de sex hörnens koordinater (rad `22`) och för cirkeln medelpunktens koordinater samt radiens längd. Hur koordinaterna definieras förklaras nedan:

Koordinatsystem för geometriska figurer

För att kunna rita geometriska figurer och placera dem behöver vi ange deras storlek och position, vilket förutsätter ett koordinatsystem på den grafiska ritytan. Följande koordinatsystem är automatiskt definierat i alla grafiska miljöer: Origo dvs positionen $(0, 0)$ är placerad i fönstrets övre vänstra hörn. x-koordinaten växer i horisontell led åt höger och y-koordinaten i vertikal led nedåt:



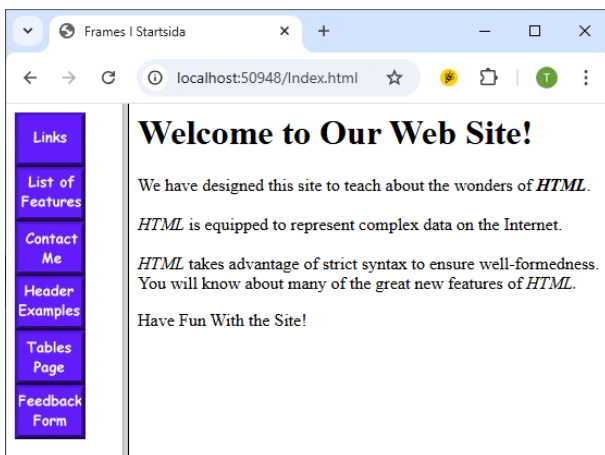
Denna bild borde man ha i minnet när man arbetar med koordinater. Enheten är en *pixel* som står för *picture element*. En pixel är en digital bilds minsta komponent – datorgrafikens atom så att säga. Pixelns faktiska storlek är beroende av den aktuella tekniska utrustningen dvs bildskärmen och dess upplösning. Vill vi placera en punkt i koordinatsystemet ovan anges punktens x-koordinat som antalet pixlar som den är borta från formens vänstra kant. Punktens y-koordinat anges som antalet pixlar som den är borta från formens övre kant.

3.9 Framesets

```
1 <!-- Index.html -->
2 <head>
3   <title>Frames I Startside</title>
4 </head>
5 <frameset cols = "110, 450">      <!-- 2 frames skapas med storlekar -->
6   <frame name = "leftframe" src = "nav.html"> <!-- Sidan nav visas -->
7   <frame name = "mainframe" src = "main.html"> <!-- Sidan main visas -->
8 </frameset>
```

Elem. frameset

Scriptet `Index` producerar i kombination med andra script som vi återkommer till, sidan till höger. Först skapar elementet `frameset` på rad 5 två frames, dvs delar webbsidan i två kolumner, den ena av storleken **110** och den andra av **450** pixlar. Startsidan till höger består av två frames som bildar ett frameset. Med navigeringsmenyn i den vänstra ramen kan man komma till webbsidans andra delar.



Elementet frame

Varje frame får med elementet `frame` ett namn och ett innehåll på raderna 6 och 7. Den vänstra ramen med namnet `leftframe` fylls med innehållet från scriptet `nav`, medan i den högre med namnet `mainframe` visas scriptet `main`. Båda script lagras som HTML-filer i samma mapp som scriptet `Index`, se nästa sida. Medan `main.html` innehåller endast vanlig text som visas ovan har filen `nav.html` lite mer intressant innehåll. Vi ska titta lite närmare på koden som återges på nästa sida.

Navigeringsmeny i en frame

Redan i scriptet `Nav` på sid 30 har vi introducerat en navigeringsmeny vars körresultat visas på sid 31. Koden har anpassats till att fungera här som navigeringsmeny i den vänstra ramen av scriptet `Index`.

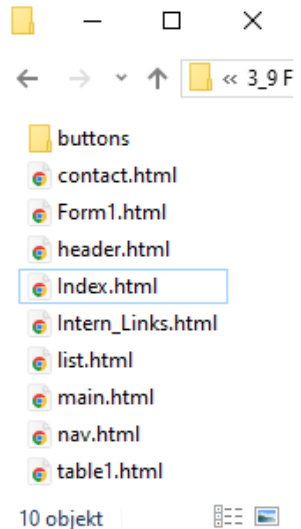
Meningen med uppdelningen av dokumentet i två frames är att efterlikna designen i verkliga webbsidor där man kan navigera till olika delar av webbsidan.

```
1 <!-- nav.html -->
2 <head>
3   <title>Navigeringsmenyn i scriptet Index</title>
4 </head>
5 <body>
6   <p>
7     <a href = "Intern_Links.html" target = "mainframe">
8       <img src = "buttons/links.jpg" width = "65" height = "50">
9     </a><br>
10    <a href = "list.html" target = "mainframe">
11      <img src = "buttons/list.jpg" width = "65" height = "50">
12    </a><br>
13    <a href = "contact.html" target = "mainframe">
14      <img src = "buttons/contact.jpg" width = "65" height = "50">
15    </a><br>
16    <a href = "header.html" target = "mainframe">
17      <img src = "buttons/header.jpg" width = "65" height = "50">
18    </a><br>
19    <a href = "table1.html" target = "mainframe">
20      <img src = "buttons/table.jpg" width = "65" height = "50">
21    </a><br>
22    <a href = "Form1.html" target = "mainframe">
23      <img src = "buttons/form.jpg" width = "65" height = "50">
24    </a><br>
25  </p>
26 </body>
```

Största nyheten i den här versionen jämfört med scriptet **Nav** på sid 30 är attributet **target** som lagts till i alla ankare, och som placerar resultatet av musklickningen på resp. bild i navigeringsmenyn i den frame som bär namnet **mainframe**, se scriptet **Index**, rad 7. Meningen med detta är att sidan som bilden man klickar på, hänvisar till, visas i den högre (större) ramen av startsidan, utan att det öppnas en ny flik, samtidigt som navigeringsmenyn finns kvar. Den ska ju användas för att navigera till andra delar av webbsidan.

Webbsidan består av 9 html-filer och mappen **buttons** som innehåller alla bilder, se bilden till höger.

Alla koder kan inte visas här. Hela mappen är zip-pad till filen [Framesets.zip](#) som kan laddas ned från kursens hemsida, se den aktuella lektionens [övningar](#).



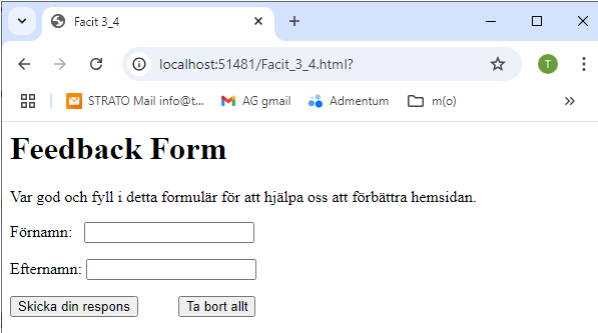
- 3.1 Kör scriptet **Tabell1** (sid 47) i din webbläsare. Ladda scriptet i din favorit editor. Ändra värdet till **table**-attributet **border** till **0, 2, 3, ...** för att se andra typer av kantlinjer. Ändra även värdet till attributet **width**. Blir tabellens bredd den procentuella delen av din webbläsares bredd?
- 3.2 Vidareutveckla scriptet **Tabell1** (sid 47) genom att lägga in ytterligare en rad till tabellen som beskriver en valfri frukt. Ändra även det totala priset i tabellens fot som en konsekvens av tillägget.
- 3.3 Modifiera scriptet **Tabell12** (sid 49) genom att ändra koden, så att körresultatet ser ut så här:

Modifierad utvecklad tabell

Så här ser en mer utvecklad tabell ut:

	Lamor är kameldjur		
	Uppdaterat v43/2024		
Producerar ull	# Pucklar	Ursprungsregion	Spotta?
Kameler (baktriska)	2	Afrika/Asien	Lama
Lamor	1	Anderna (Sydam.)	

- 3.4 Lägg till scriptet **Form1** (sid 51) en textbox till som ska innehålla efternamnet. Ändra även koden så att formen får det här utseendet:



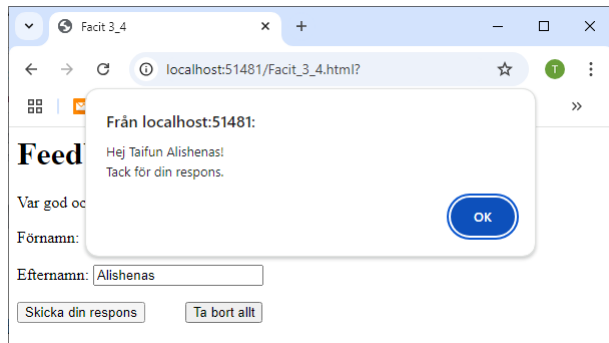
Feedback Form

Var god och fyll i detta formulär för att hjälpa oss att förbättra hemsidan.

Förnamn:

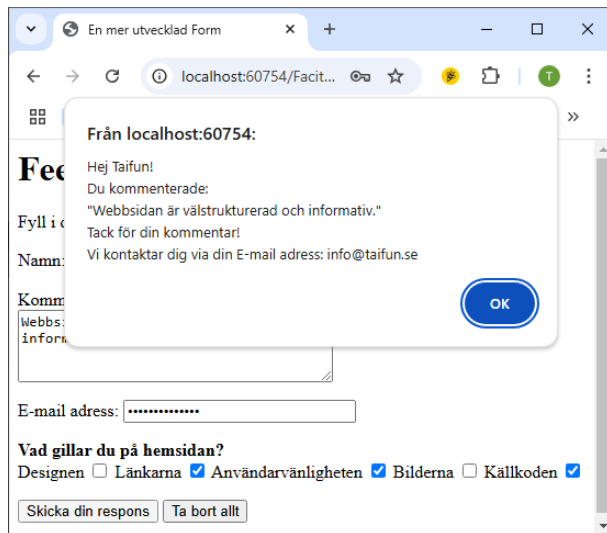
Efternamn:

Man ska sedan fylla i sitt för- och efternamn och få följande svar när man klickar på knappen Skicka din respons:



För att få detta svar måste du även ändra koden i funktionen `showData()`.

3.5 Vidareutveckla scriptet **Form2** (sid 53) så att formen får det här utseendet:



Dvs modifiera koden, så att JavaScript funktionen `showData()` hämtar data inte bara från textboxen `Namn` (rad 9), utan även från multiline textarean `Kommentar` (rad 15) och från textboxen `E-mail adress` (rad 22). För att uppnå resultatet ovan, bör du modifiera inte bara scriptets HTML-kod utan även JavaScript funktionen.

Intressant blir att se hur den maskerade E-mail adressen ”avslöjas” i JavaScripts *alert box* (sid 52).

3.6 Modifiera scriptet **Form3** (sid 53) i följande punkter:

- 1) Minska analet alternativ i rullgardinsmenyn till tre.
Ge valfria namn till dina betygssteg.
- 2) Döp om radioknappen Referens i en bok till Tips från en kompis .
- 3) Ta bort radioknappen Taifun.se .
- 4) Döp om checkboxen Bilderna till Strukturen .
- 5) Lägg till i slutet en multiline textbox som frågar:

Hur skulle vi kunna förbättra vår hemsida? Ge oss tips:

3.7 Scriptet **Intern_Links** (sid 58) har två huvudrubriker som båda är försedda med interna länkar. Komplettera scriptet med en tredje rubrik och lista upp dina favoritämnen där:

Mina tre favoritämnen i skolan:

Definiera en intern länk som leder till denna plats. Länka från de andra rubrikerna till den nya rubriken. Länka även från denna plats till de två andra rubrikerna.

3.8 I scriptet **Picture** (sid 64) finns en länk till Taifuns mail som är definierad i **area**-elementet med attributet **shape** = "circle". Flytta i ett nytt script denna länk (hotspot) till en annan plats, närmare bestämt till fyrkanten på vänstra sidan av bilden, under texten **Home Page**.



3.9 Ladda ned zipp-filen [Framesets.zip](#) och expandera den. Bygg med dessa filer webbsidan som scriptet **Index** är startsidan för (sid 66).

Modifiera sedan sidan genom att lägga ytterligare menyer (minst en) till navigeringsmenyn med egna bilder och ev. scriptfiler. Placera dina nya moduler på korrekta platser, så att allt fungerar på ett meningsfullt sätt.

Kapitel 4

Cascading Style Sheets (CSS)

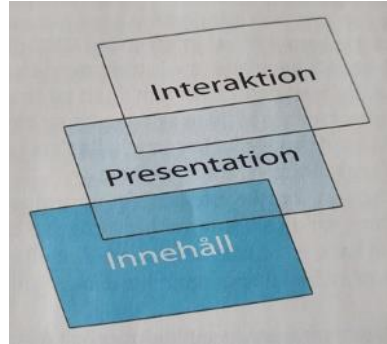
Ämne	Sida	Program
4.1 Inline Styles	72	Inline
4.2 Internal Styles	74	Declared
4.3 Conflicting Styles	75	Advanced
4.4 External Styles	77	External
4.5 Absolut positionering	79	Positioning
4.6 Relativ positionering	80	Positioning2
Övningar till kap 4	81	

4.1 Inline Styles

De tre skikten

I början av kursen i Webbutveckling 1 talade vi om de *tre olika skikten* inom Webbutveckling:

- *Innehållet* kodas med **HTML**.
- *Presentationen* formges med **CSS**.
- *Interaktionen* programmeras med **JavaScript**.



Det rekommenderades att separera de här tre skikten. I tre kapitel har vi gått igenom det första skiktet HTML som skulle kunna kallas för verktyget som ger *innehåll* men även *struktur* åt webbidor. Nu ska vi ägna oss åt det andra skiktet CSS som är tänkt att *presentera* webbsidor på bäst möjliga sätt. Här står layouten, formen, utseendet och det visuella intrycket i centrum.

Vad är CSS?

CSS är ett *stilspråk* som används för att skapa s.k. *Style Sheets*. En *sheet* betyder på engelska ett blad, alltså i datasammanhang *fil*. I denna fil skriver man kod som definierar stilregler som används för att formatera text i HTML-element. Med *formatering* menas allt som påverkar textens layout som t.ex. färger, typsnitt, storlekar, radavstånd, indrag, marginaler, bakgrundsfärger, bakgrundsbilder osv. Dessa regler fungerar på samma sätt som kommandon som används i ordbehandlingsprogram, vare sig de är av typ WYSIWYG (som t.ex. Word) eller skrivs som kod.

Historien

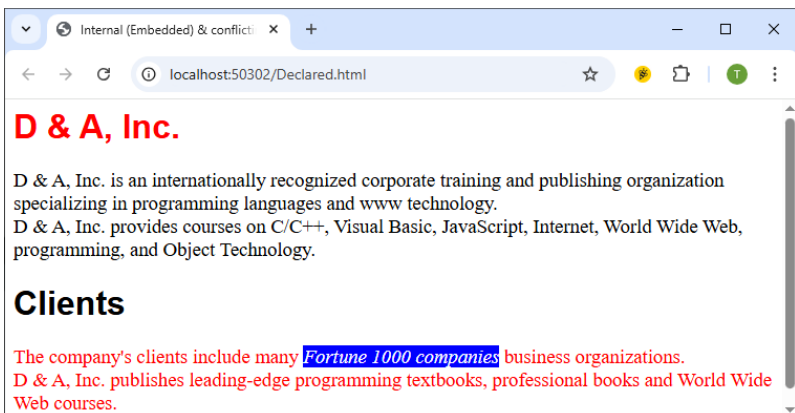
CSS introducerades redan 1994 och blev standard 1996 vilket betyder att det rekommenderades av W3C, det officiella internationella organet som tar fram standarder inom HTML. En fullständig specifikation både för HTML och CSS kan hittas på W3C:s websida.


```
1 <!-- Inline.html -->
2 <head>
3   <title>Inline Styles</title>
4 </head>
5 <body>
6   <p>Denna text har ingen formatering (style) alls på sig.</p>
7   <!-- Attributet style i ett HTML-element kan deklarera -->
8   <!-- en inline style: -->
9   <p style = "font-size: 15pt">
10    Denna text får font-storleken 15 pt på sig med värdet
11    <em>font-size: 15pt</em> till attributet style.
12  </p>
13
14  <!-- Flera styles kan separeras med ; i ett HTML-element: -->
15  <p style="font-size: 20pt; color: blue">
16    Denna text får font-storleken 20 pt med
17    <em>font-size: 20pt</em> och färgen blå med
18    <em>color: blue</em> till attributet style.
19  </p>
20
21  <p style="font-size: 25pt; color: red">
22    Styles som med attributet style byggs in i ett HTML-element
23    kallas för <em>Inline Styles</em>.
24  </p>
25
26  <p style="font-size: 25pt">
27    Andra sätt att använda Styles är <em>Internal</em> och
28    <em>External Styles</em>.
29  </p>
30 </body>
```



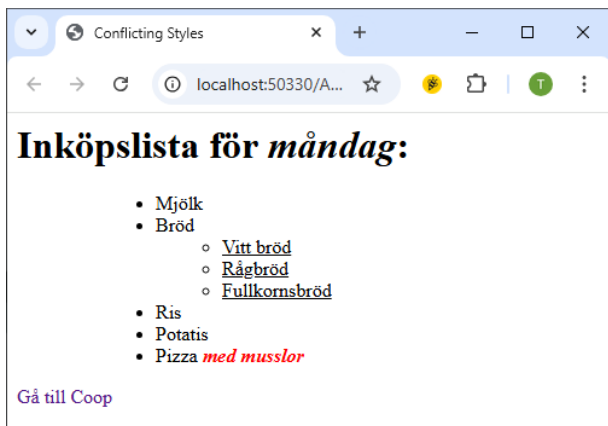
4.2 Internal Styles

```
1 <!-- Declared.html -->
2 <head>
3   <title>Internal (Embedded) & conflicting Styles</title>
4   <!-- <style>-elementet (rad 5-10) innehåller CSS-kod -->
5   <style type="text/css" /* Internal Styles deklarerar i <head> */
6     em {background-color: blue; color: white}
7     h1 {font-family: arial, sans-serif, 'Times New Roman'}
8     p {font-size: 14pt}
9     .special {color: red} /* Deklarerar en Style class (.) */
10  </style> <!-- Slut på CSS-kod -->
11 </head>
12 <body>
13   <!-- Attributet class tillämpar färgen red i h1 -->
14   <h1 class="special">D & A, Inc.</h1>
15   <p>
16     D & A, Inc. is an internationally recognized corporate
17     training and publishing organization specializing
18     in programming languages and www technology.<br>
19     D & A, Inc. provides courses on C/C++,
20     Visual Basic, JavaScript, Internet, World Wide Web,
21     programming, and Object Technology.
22   </p>
23
24   <h1>Clients</h1>
25   <p class="special" <!-- class tillämpar färgen red i p -->
26     The company's clients include many <!-- em ärver p:s storlek -->
27     <em>Fortune 1000 companies</em> business organizations.<br>
28     D & A, Inc. publishes leading-edge programming
29     textbooks, professional books and World Wide Web courses.
30   </p> <!-- Conflicting styles: em ärver p:s color (red), men -->
31 </body> <!-- överskuggar(overrides) den p.g.a. högre specificitet -->
```

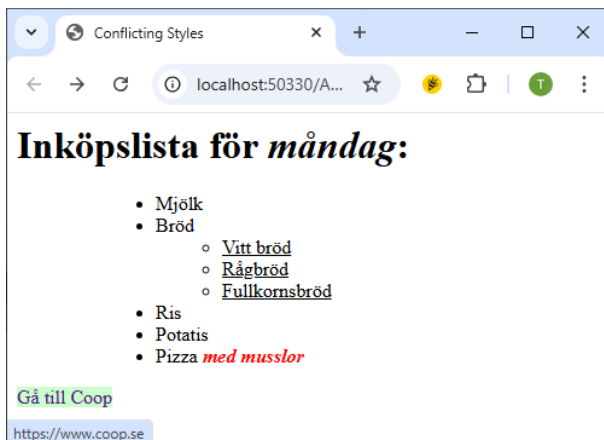


4.3 Conflicting Styles

```
1 <!-- Advanced.html -->
2 <head>
3   <title>Conflicting Styles</title>
4   <style>
5     a.nodect{text-decoration: none}           /* Används i a */
6     a:hover{background-color: #ccffcc}       /* Pseudoklass anv. i a */
7     li em {color: red; font-weight: bold}    /* Används på rad 25 */
8     ul {margin-left: 75px}                   /* Används på rad 14 */
9     ul ul {text-decoration: underline; margin-left: 15px}
10  </style>                                     <!-- Används på rad 17 -->
11 </head>
12 <body>                                     <!-- Conflicting Styles överskriver HTML-element: -->
13 <h1>Inköpslista för <em>måndag</em>:</h1> <!-- em no conflict -->
14 <ul>                                       <!-- Conflicting Style ul -->
15   <li>Mjölk</li>
16   <li>Bröd
17     <ul>                                     <!-- Conflicting Style ul ul -->
18       <li>Vitt bröd</li>
19       <li>Rågbröd</li>
20       <li>Fullkornsbröd</li>
21     </ul>
22   </li>
23   <li>Ris</li>
24   <li>Potatis</li>
25   <li>Pizza <em>med musslor</em></li>      <!-- Conflicting -->
26 </ul>                                       <!-- Style li em -->
27 <p><a class = "nodect"
28   href = "https://www.coop.se">Gå till Coop</a></p>
29 </body>
```



Mus över [Gå till Coop](#) :



The screenshot shows a web browser window with the title "Conflicting Styles". The address bar displays "localhost:50330/A...". The main content area features a heading "Inköpslista för *måndag*:" followed by a bulleted list of items: "Mjök", "Bröd" (with sub-items "Vitt bröd", "Rågbröd", and "Fullkornsbröd"), "Ris", "Potatis", and "Pizza *med musslor*". Below the list is a button labeled "Gå till Coop" and a URL "https://www.coop.se".

Conflicting Styles

localhost:50330/A...

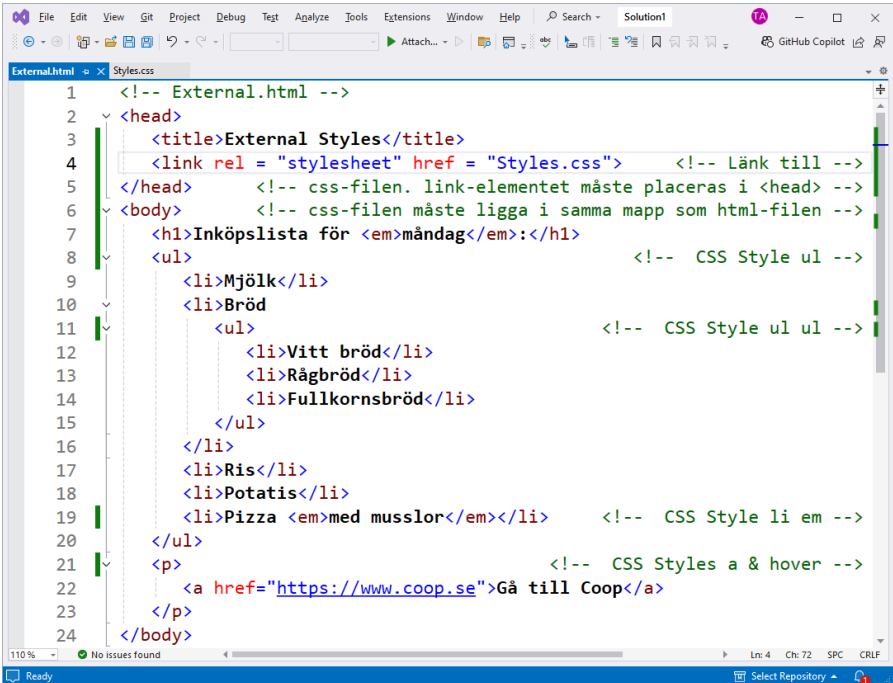
Inköpslista för *måndag*:

- Mjök
- Bröd
 - Vitt bröd
 - Rågbröd
 - Fullkornsbröd
- Ris
- Potatis
- Pizza *med musslor*

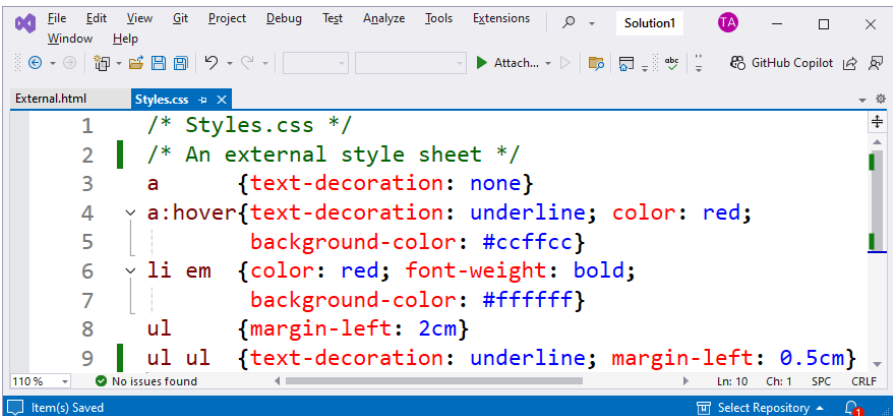
[Gå till Coop](#)

<https://www.coop.se>

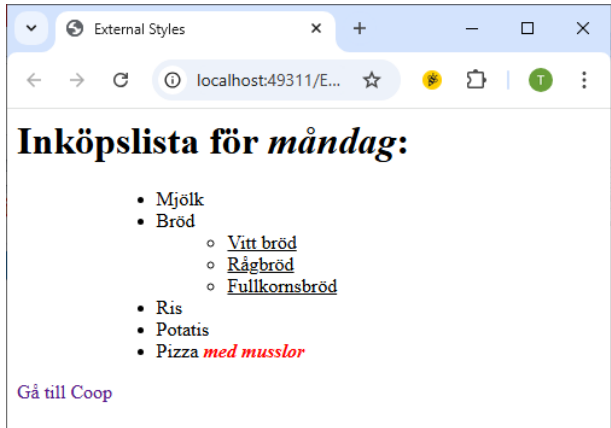
4.4 External Styles



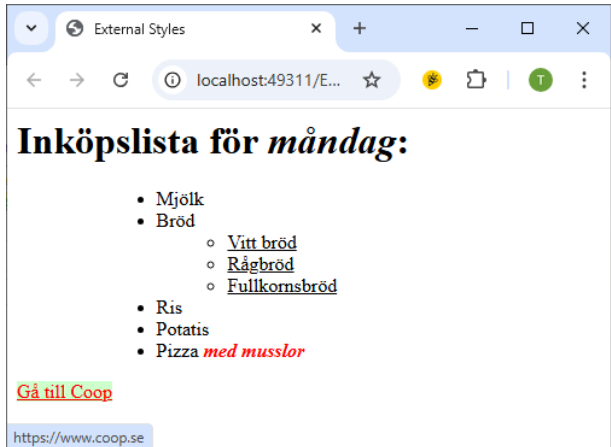
```
1 <!-- External.html -->
2 <head>
3   <title>External Styles</title>
4   <link rel = "stylesheet" href = "Styles.css" <!-- Länk till -->
5 </head> <!-- css-filen. link-elementet måste placeras i <head> -->
6 <body> <!-- css-filen måste ligga i samma mapp som html-filen -->
7   <h1>Inköpslista för <em>måndag</em>:</h1>
8   <ul> <!-- CSS Style ul -->
9     <li>Mjök</li>
10    <li>Bröd
11      <ul> <!-- CSS Style ul ul -->
12        <li>Vitt bröd</li>
13        <li>Rågbröd</li>
14        <li>Fullkornsbröd</li>
15      </ul>
16    </li>
17    <li>Ris</li>
18    <li>Potatis</li>
19    <li>Pizza <em>med musslor</em></li> <!-- CSS Style li em -->
20  </ul>
21  <p> <!-- CSS Styles a & hover -->
22  <a href="https://www.coop.se">Gå till Coop</a>
23  </p>
24 </body>
```



```
1 /* Styles.css */
2 /* An external style sheet */
3 a {text-decoration: none}
4 a:hover{text-decoration: underline; color: red;
5   background-color: #ccffcc}
6 li em {color: red; font-weight: bold;
7   background-color: #ffffff}
8 ul {margin-left: 2cm}
9 ul ul {text-decoration: underline; margin-left: 0.5cm}
```



Mus över [Gå till Coop](#) :



4.5 Absolut positionering

```
1 <!-- Positioning.html -->
2 <head>
3   <title>Absolut positionering</title>
4 </head>
5 <body>
6   <p><img src = "i.gif" style = "position: absolute;
7     top: 0px; left: 0px; z-index: 1"
8     alt = "Positionerad bild 1"></p>
9
10  <p style = "position: absolute; top: 50px; left: 15px;
11    z-index: 3; font-size: 20pt;">Absolut positionering</p>
12
13  <p><img src = "circle.gif" style = "position: absolute;
14    top: 25px; left: 100px; z-index: 2"
15    alt = "Positionerad bild 2"></p>
16 </body>
```

Attributet **position**

Före CSS var det svårt att kontrollera positioneringen av element i ett HTML-dokument. Det var webbläsaren som ofta bestämde positioneringen. CSS introducerade attributet **position** som ger utvecklaren kontrollen över positioneringen av element, genom att bl.a. bjuda på möjligheten att använda *absolut positionering*.



Scriptet **Positioning** (ovan) använder två bilder, *i.gif* och *circle.gif*, och en text för att sätta ihop dem till dokumentet ovan. Raderna **6-8** placerar den första bilden (*i.gif*) så att den hamnar, oberoende av den normala ordningen i koden, på den absoluta positionen **0** pixlar från den vänstra och övre kanten.

Attributet **z-index** ordnar överlappningen av elementen. Element med högre **z-index** placeras framför element med lägre **z-index**. I vårt exempel har bilden *i.gif* det lägsta **z-index**et, nämligen **1**. Därför hamnar den i bakgrunden. *circle.gif* har index **2** och texten index **3**. Om man inte anger ett värde till **z-index** placeras elementen från bakgrunden framåt i den ordning de skrivs till dokumentet,

4.6 Relativ positionering

```
1 <!-- Positioning2.html -->
2 <head>
3   <title>Relativ positionering</title> <!-- Relativ till -->
4   <style>                               /* andra element */
5     p      {font-size: 1.3em; /* 1em = bredden på M */
6             font-family: verdana, arial, sans-serif}
7     .super {color: red; font-size: 0.8em; height: 1em}
8     .sub   {position: relative; top: -1ex}
9     .shiftleft {position: relative; left: -1ex}
10    .shiftright{position: relative; right: -1ex}
11  </style>                               <!-- 1ex = höjden på x -->
12 </head>
13 <body>
14   <p>Texten i slutet av denna mening är
15   <span class="super">upphöjd (in superscript)</span>.</p>
16
17   <p>Texten i slutet av denna mening är
18   <span class = "sub">nedsänkt (in subscript)</span>.</p>
19
20   <p>Texten i slutet av denna mening är
21   <span class = "shiftleft">förskjuten åt vänster
22   (shifted left)</span>.</p>
23
24   <p>Texten i slutet av denna mening är
25   <span class="shiftright">förskjuten åt höger
26   (shifted right)</span>.</p>
27
28 </body>
```



- 4.1 Kör scriptet **Inline** (sid 73) i din webbläsare. Ladda scriptet i din favorit editor. Ändra värdena till attributet **style** så att texterna skrivs i omvänd fontstorlek, dvs den största fonten först och den minsta sist. Ändra även värdena till attributet **color** genom att experimentera med andra färger.

Byt ut färgvärdena **blue** och **red** mot färgkoder genom att ta reda på koder-na. T.ex. är **#0000ff** färgkoden till **blue**.

- 4.2 Modifiera scriptet **Declared** (sid 74) genom att ändra namnet till Style-klassen **special** till ditt namn. Testa om scriptet fungerar precis som förut. Det borde det göra om du ändrat korrekt. Kasta om ordningen av fonterna i Stylen **h1** och testa. Vilken slutsats drar du?

Ta bort attributet **color** från Stylen **em**. Vilken färg får den framhävda texten? Från vilket element ärver texten sin färg?

Byt även ut den framhävda textens bakgrundsfärg till andra färger. Experimentera både med färgnamnen och färgkoder.

- 4.3 Mata in scriptet **Advanced** (sid 75) i din favorit editor och kör det i din webbläsare. Gå med musen över texten **Gå till Coop**. Vilken Style i scriptet ändrar textens bakgrundsfärg? Ändra bakgrundsfärgen till **red**. Vilken Style i scriptet är ansvarig för indragningen av item **Mjölk**? Ändra indragningen till 50 pixlar. Ändra även indragningen av item **Vitt bröd** till 10 pixlar.

Byt ut färgen av texten **med musslor** till **blue** och låt den inte längre skrivas i fet utan i normal stil. Experimentera med ytterligare stiländringar i scriptet. Varför har texterna **måndag** och **med musslor** olika formateringar, fast båda ligger i koden i HTML-elementet ****?

- 4.4 Dela upp scriptet **Declared** (sid 74) i två filer genom att flytta den interna css-koden till en extern css-fil, så att alla internal Styles blir external Styles.

Först ska scriptet generera samma dokument som det ursprungliga scriptet **Declared** (sid 74) gjorde.

Modifiera sedan scriptet genom att göra de ändringar som föreslås i övn. 4.2 (Övningar 29).

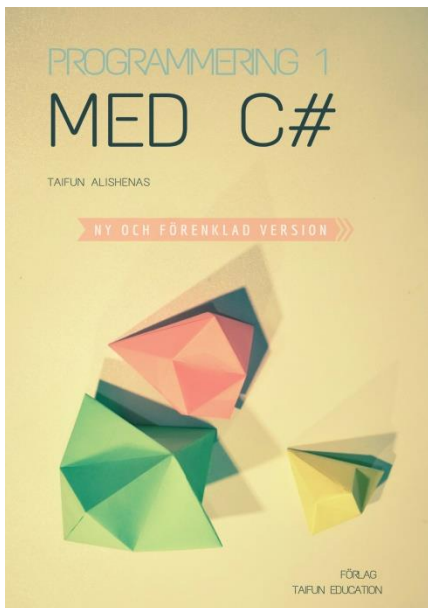
- 4.5 Läs igenom texten och gå igenom koden på sid 79. Skriv ett script som positionerar elementen i scriptet **Positioning** så att hela bildgruppen hamnar i det nedersta högre hörnet av browserfönstret på sid 79.

4.6 Dela upp scriptet **Positioning2** (sid 80) i två filer genom att flytta all css-kod till en extern css-fil.

Först ska scriptet generera samma dokument som det ursprungliga scriptet **Positioning2** gjorde.

Modifiera sedan scriptet genom att göra experiment med de relativa positioneringarna, t.ex. med font-storlekarna, textförskjutningarna osv.

Programmering 1 med C#



Ur innehållet:

Grundbegrepp i programmering
Datatyper, variabler & tilldelning
Utskrift till grafisk miljö
Windowsprogrammering
C# Console & Windows Applications
Interaktiva grafiska gränssnitt
Kontrollstrukturer
Klasser, objekt och referenser
Metoder
Rekursiva metoder
Sammansatta datatyper: Arrays
Dynamiska arrays: Listor
Sökning & sortering
Kryptering av text
Hantering av slumpstal
Undantagshantering
Vad är objektorienterad programmering?
Installation av Visual Studio.NET
Konfiguration av Visual Studio.NET
Projekt i Visual Studio.NET
Övningar & projektuppgifter
Fullständiga lösningar till övningar

www.taifun.se

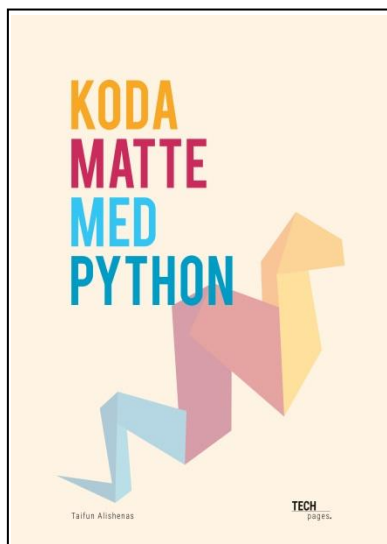
Koda matte med Python

Programmering i matematik

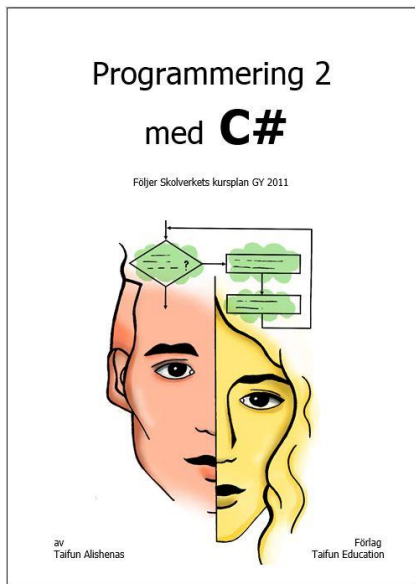
En enkel, pedagogisk lärobok som kompletterar matematikundervisningen med inslag av programmering. Den vägleder både lärare och elever genom att kombinera teori med praktiska övningar och fullständiga lösningar.

Boken presenterar ett pedagogiskt koncept om hur programmering kan integreras i kurserna Matematik 1 och Matematik åk 7-9.

Ladda ned gratis smakprov.



Programmering 2 med C#



Ur innehållet:

Windowsprogrammering
Grafiskt gränssnitt mot Internet (webbläsare)
Grafiskt gränssnitt med menyval
Multiple Document Interface
Objektorienterad programmering
Objektorient. modellering & implementation
Metoder i OOP / Generics
LINQ / Lambdauttryck
Delegater / Metodgrupper
Arv och polymorfism
Abstrakta klasser & metoder
Virtuella metoder
Filhantering / Slumplösenord
Kryptering av filer / Tabellhantering i filer
Databaser / Relationsdatabasmodellen
Introduktion till SQL databaser
Visual Studios SQL-Server
Grafiskt gränssnitt mot databasen
En SQL-klient i C#
Att skapa och designa en databas
Databas med egna funktionaliteter
Projektuppgifter & övningar
Fullständiga lösningar till alla övningar

Utveckla en egen webbläsare (ex. ur boken ovan):

webbapp samt till [Android](#) och [iOS](#).' and three download buttons: 'Get it on Webbapp', 'GET IT ON Google Play', and 'Download on the App Store'. On the right side of the browser window, there is a smartphone displaying the 'Mattekollen' app interface. The app shows a title '1.3 Potenser', a large '2^3' with a red arrow pointing to '3' labeled 'Exponent' and a blue arrow pointing to '2' labeled 'Bas'. Below this are three text boxes: a pink one for 'Potens med positiv exponent: 2^3 = 2 · 2 · 2 = 8', a red one for 'Potens - upprepad multiplikation: ev. 2 med sig själv, 3 gånger.', and a green one for 'Potens med negativ exponent: 2^-3 = 1/2^3 = 1/8'. At the bottom of the app screen, it says 'Allt "invertera" lex. 10 ger 1/100.' and '© 2012 Taifun Education'."/>

Programmering i matematik

Tio lektioner

Ett läromedel som integrerar programmering i matematikundervisningen.

Kan användas för självständigt arbete i klassrummet eller på distans.

Kräver inga förkunskaper i programmering.

För gymnasiets kurser i Matematik 1 (a, b, c) och för högstadiets åk 7-9.

Ur innehållet

Varför är såpbubblor runda?

Eftersom de följer naturens lag och antar den minst möjliga ytan vid samma volym. Detta kan uppnås endast som klot (sfär), en geometrisk figur som saknar hörn och är dessutom vacker.

Naturen minimerar energin. Effektiviteten möter estetiken.

Genom att kombinera programmering med matematik kan du lyfta hemligheterna bakom samma naturlag som gör såpbubblorna runda.



Koda direkt i vår mobila pythonmiljö. Ladda ned appen *Mattekollen*.