15 November

Uppgifter

Observera att kursens syfte är att ni skall lära er tekniken. De sidor ni producerar kommer att bedömas med avseende på vilka tekniska lösningar ni gör i förhållande till kursmål och med avseende på respektive uppgifts syfte. Så kallad ”kontent” (text, bilder, ljud och animationer) lämnas utanför bedömningen, likaså färgval, estetisk komposition.

Tennisklubb – individuell

Vi arbetar vidare med våra tennisklubbar, nästa steg efter att ha fått webbplatsen responsiv; efter att ha sett till att den anpassar sig automatisk till olika fönsterstorlekar, är att få in ett menysystem. Lite beroende på din design kan det passa med en traditionell rullgardinsmeny, någon form av popup-meny eller kanske en modern hamburgarmeny. Vill du vara flexibel kan du se om du kan göra dina menyer responsiva så att du har olika sorters meny beroende på skärmens bredd.
Projekten nedan publiceras lämpligen på Github (eller liknande tjänst) så att läraren enkelt kan följa ert arbete.

Spel/interaktiva sidor – individuell

För att fördjupa kursmålet om interaktivitet och dynamik finns det möjlighet att även arbeta med ett enklare spel. Några förslag är: Snake, Pong, Sudoku, Tre i rad; men din fantasi sätter gränserna här. Det går naturligtvis att hitta dynamiska och interaktiva applikationer som inte är spel, ritprogram, miniräknare, måttomvandlare, kom-ihåg-listor; eller du kanske kan hitta på någonting annat som intresserar dig.
Med en HTML Canvas får du en rityta för att att skapa spelbräden och annat. Kombinerar du detta med JavaScript kan du få ”liv och rörelse”.
Använder du HTML Web Storage kan du lagra flera megabyte lokalt i webbläsaren mellan gångerna.
https://www.w3schools.com/html/html5_canvas.asp
https://www.w3schools.com/graphics/game_intro.asp
https://www.w3schools.com/html/html5_webstorage.asp

Samåkningswebb – grupp

Med stigande bränslepriser och med en klimatkris som blir allt mer påtaglig vill vi skapa en sida för att dels öka intresset för samåkning, ni bör alltså ha någon form av informationssida (även om innehållet på den sidan inte kommer att bedömas), vidare skall det gå att registrera intresserade bilägare, intresserade resenärer. När någon registrerat sig som ”ägare” eller ”passagerare” så kan hen i nästa steg antingen anmäla enstaka resor eller pendlingsresor. Exempel ”på måndag 1/3 åker jag från Mjölby till Töre, vi åker senast klockan 6:30 på morgonen och vi planerar tre halvtimmespauser under resan. Jag har plats för en vuxen fram och ett mindre barn i baksätet, eftersom vi har både packning och en grand danois så har vi begränsat utrymme för ytterligare bagage och den medföljande hunden gör att resan nog inte passar pälsdjursallergiker” eller ”vi är två storvuxna män, 19 och 25 år, som vill åka till Köpenhamn från Skänninge under sista veckan i januari”. Eller pendlingsresor ”jag vill åka med från Norrköping till Linköping måndag till fredag, jag behöver vara i Linköping före 7 på morgonen och jag vill helst inte åka hem senare än klockan 16:30”. Bilägare kan registrera vad önskad ersättning och resenärerna kan registrera vad de är beredda att betala. Även andra saker kan vara intressanta, kanske bilägaren vill ha resesällskap som kan hjälpa till med körningen om det är en långresa (körkort..), resenären kanske inte vill åka i fordon där det brukar transporteras pälsdjur eller så kanske hen vill varna för att åksjuka kan förekomma vilket tenderar kräva oplanerade pauser.

Sammanfattningsvis är det här en lite mer omfattande uppgift än ”tennisbanan” och ni har möjlighet att utveckla den i lämpliga riktningar som passar gruppmedlemmarnas intressen.
Sammåkningstjänsten skall naturligtvis ha validering av inmatad data och den bör vara responsiv så att även resenärer utan laptop skulle kunna klara av att använda tjänsten direkt från sina mobiler utan problem.
Det kanske är intressant med kartor, positioner och geografi på en resesida, kolla här om ni är nyfikna på att utforska sådana saker:

https://www.w3schools.com/html/html5_geolocation.asp

Ni kan säkert hitta intressanta saker att göra med exempelvis Hitta, Eniro och Google, för att få kartor.

Felsökning – muntlig presentation

Enligt kursplanen skall ni redovisa ett antal kunskapsmål relaterade till felsökning. Eftersom ni skall kunna nå alla tre betygsnivåerna här (IG/G/VG) krävs en lite mer omfattande redovisning. Mitt förslag är att vi bokar in tider för individuella samtal vid kursens slut, ämnets natur är ju sådant att ni måste ha hunnit öva på att felsöka saker innan ni kan berätta hur ni gör. Så gott jag kan se har ni då chansen att dels berätta om hur ni genomfört de olika projekten som är kursens praktiska moment i övrigt, detta då speciellt ur felsökningshänseende. Sedan får ni ju även möjlighet att relatera till andra situationer där ni felsökt hemsidor och liknande som ni arbetat med i olika sammanhang.

Instudering

Git

Förra veckan skaffade ni konton på Github och installerade Github Desktop. Nu skall vi bekanta oss med git från kommandoraden. Ni kan naturligtvis installera plugin för git i VS Code och liknande, vilket säkert underlättar det dagliga arbetet, men då kommer vi in på individuella arbetsflöden och verktygsval så detta lämnar vi utanför kursen. Själv skriptar jag gärna repetitiva arbetsflöden, så att jag sällan behöver köra kommandon direkt för sådana uppgifter som jag gör ofta och som kräver flera steg.

Hämta först git här:

https://git-scm.com/downloads

sedan installerar ni.

En snabbkurs om git hittar ni här:

https://www.w3schools.com/git/default.asp?remote=github

Den officiella handboken finns att läsa här (eller att ladda ned):

https://git-scm.com/book/en/v2

Det finns många lathundar på nätet, men den här tycker jag är trevlig:

https://www.hostinger.com/tutorials/basic-git-commands

Innan du sätter igång är det några saker du måste känna till, för det första går det utmärkt att köra git separat utan server eller andra inblandade. Fördelen med detta är att du snabbt kan rulla tillbaka om du exempelvis testat någonting som inte fungerade. Du kan skapa olika grenar för olika versioner vilket kan vara överraskande praktiskt även på småprojekt.
Skall du arbeta med andra måste du identifiera dig och det kan vara lämpligt att ställa in en epostadress.

```
git config ––global user.name ”Ditt Namn”

git config ––global user.email ”din@epost.net”
```

Eftersom git inte kan gissa måste du tala om vilka filer som skall vara med i projektet, men först måste du skapa ett projekt:

Kör följande i den katalog du har dina arbetsfiler:

```
git init
```

För att sedan tala om vilka filer som skall vara med:

```
git init min.nya.fil
```

Ibland kan det vara praktiskt att tala om vilka filer som inte skall vara med, framför allt brukar det röra sig om olika temporärfiler som exempelvis editorer och kompilatorer brukar fylla våra diskar med. Dessa listar du i en fil som du kallar .gitignore

https://www.w3schools.com/git/git_ignore.asp

Oavsett var du är och oavsett vad du tänker göra i nästa steg så är följande kommando ofta användbart:

```
git status
```

När du sedan arbetat klart i dina filer kör du en commit:

```
git commit -m ”en bra förklaring som berättar syftet med arbetet du gjort”
```

då läggs dina filer till i ditt repo, så att de dels finns med men även så att du kan skicka dem (vid behov) till en server, kanske github..
Det är viktigt att din förklarande text berättar varför du gjort det du gjort, git kan ju lätt visa vad du gjort, men varför vet du bara själv, om du inte skriver in det när du commitar din kod.

Du grenar av en ny gren med hjälp av branch:

```
git branch ”grenens namn”
```

Skall du hoppa mellan grenarna:

```
git checkout ”namn på en annan gren”
```

Skall ni sedan ladda upp koden på github:

```
git push origin
```

Vill du tanka hem ett projekt från github:

```
git clone ”url till projektet”
```

har du redan tankat hem ett projekt men vill se till att du har de senaste förändringarna:

```
git pull
```

Pull är en kombination av fetch och merge.

CSS

Ni har redan arbetat med CSS sedan en tid men det är några saker vi måste förtydliga. För det första att CSS handlar om utseende och placering. Att det står för Cascading Style Sheets betyder att det kan finnas flera CSS-regler, eller ”stilmallar/mallar” (kärt barn..) som påverkar ett specifikt objekt. Den sista regeln innan objektet vinner, ifall det finns flera regler för samma sak som appliceras på samma objekt.
Reglerna å sin sida kan peka på en typ av html-tagg, en klass (som du alltså kan definiera själv när du vill göra regler som hanterar en grupp taggar) eller ett id (egendefinierat namn som du sätter på en specifik tag du vill påverka med en regel). Det sista fallet går även att lösa genom att lägga in CSS direkt i en tag. Exempel:

```
<p style=”color: red;”>
```

så får du ett rött stycke. När du skriver en stilregel för en html-tag, exempelvis <p> ser det ut såhär

```
p {
  color: red;
}
```

Arbetar du i stället med klasser, som kan definieras såhär

```
<p class=”SYNE22LIN”>
```

Så skriver du din stilmall såhär

```
.SYNE22LIN {
  color: red;
}
```

Om du i stället definierat en id blir det såhär:

```
<p id=”maria”>

#maria {
  color: red;
}
```

Fördelen med id framför att definiera stilmallen direkt i taggen är att du då enkelt kan kombinera samma sida med flera olika stilmallar efter behov.

https://www.w3schools.com/css/css_selectors.asp

För att använda CSS från din html använder du någon av följande metoder:

Direkt i en tagg:

```
<p style=”color: blue;”>
```

I en separat fil (som kan återanvändas till flera olika sidor vars stil ändras enkelt i en enda fil):

```
<link rel="stylesheet" href="styles.css">
```

Eller i ett separat block i din html-fil:

```
<style>
p    {color: red;}
</style>
```

Behöver du ändra stilen på ett objekt på en sida kan du även komma åt den från JavaScript via DOM:

https://www.w3schools.com/xml/dom_intro.asp

Några praktiska saker att piffa upp din sida med:

Placering av objekt är ofta besvärlig, ibland kan man fuska med att använda en tabell i html, men för att få bättre kontroll:

https://www.w3schools.com/css/css_positioning.asp
https://www.w3schools.com/css/css_align.asp
https://www.w3schools.com/css/css_padding.asp
https://www.w3schools.com/css/css_margin.asp
https://www.w3schools.com/css/css_border.asp
https://www.w3schools.com/css/css3_flexbox.asp

Ofta jobbar vi med grupper av saker på sidan, då vill vi ofta ha visuella markörer som talar om att dessa saker hänger ihop, det kan vi åstadkomma med följande:

https://www.w3schools.com/css/css_boxmodel.asp
https://www.w3schools.com/css/css_outline.asp
https://www.w3schools.com/css/css_combinators.asp

Om du behöver jobba med individuella saker som kan variera mellan olika objekt på skärmen, exempelvis länkar som man besökt jämfört med länkar man ännu inte besökt, element som har fokus när användaren tabbar runt eller när musen hovrar, pseudoklasser heter vår hjälpreda här:

https://www.w3schools.com/css/css_combinators.asp

När vi skall skapa menyer av olika slag kan en navigation-bar eller en drop-down vara intressant:

https://www.w3schools.com/css/css_navbar.asp
https://www.w3schools.com/css/css_dropdowns.asp

Behöver du visa att du är en modern webbdesigner måste du naturligtvis hitta på finurliga animationer och sådana kan du göra direkt i CSS utan att gå omväg via JavaScript:

https://www.w3schools.com/css/css3_animations.asp
https://www.w3schools.com/css/css3_2dtransforms.asp
https://www.w3schools.com/css/css3_3dtransforms.asp

https://www.w3schools.com/css/css3_transitions.asp

Samma sak med tipsbubblor:

https://www.w3schools.com/css/css_tooltip.asp

Koolt ögongodis:

https://www.w3schools.com/css/css3_gradients.asp
https://www.w3schools.com/css/css3_shadows.asp
https://www.w3schools.com/css/css3_masking.asp

Till sist får vi komma ihåg att det finns matematiska funktioner i CSS, vilket kan vara praktiskt när du vill placera olika saker i förhållande till varandra och i förhållande till skärmens geometri:

https://www.w3schools.com/css/css_math_functions.asp

Min förhoppning är att tipsen ovan skall inspirera till roliga sidor och fortsatta övningar.