**LESS**

**Introduktion**

LESS är ett verktyg för att skapa mer funktionalitet till CSS.

Filer med LESS-kod har tillägget .less

Det är en s.k. preprocessor vilket innebär att koden först omvandlas till vanlig CSS som webbläsaren förstår.

Ungefär som att PHP omvandlas till HTML när det skickas till webbläsaren.

**Winless**

Man kan lägga in länkar till en Javascript-fil som omvandlar .less till CSS och får då också lägga in en länk till .less-filen. Detta är dock inte bra när det gäller en ”skarp” sida eftersom det blir långsammare. Vi ska istället omvandla .less-kod till CSS. Då kan man länka till CSS precis som innan.

Winless är ett verktyg för att omvandla LESS-kod till CSS.

Hämta det på <http://winless.org/>

Gör så här:

* Skriv less-kod och lägg det i en mapp.
* Lägg till den mappen i Winless.
* När du har lagt till en mapp ska du inte behöva göra något mer med Winless.
* När du gör ändringar i din .less-kod och sparar känner Winless av detta och omvandlar till CSS.

**Övningar**

* Gör ett repository och kalla det LESS eller något.
* Hämta indexfilen och less-filen som ligger på Yammer.
* I indexfilen finns en länk till vanlig css.
* Vi behöver inte ha någon koppling till CSS-filen.
* Ta mappen som heter css och kopiera den till Winless eller leta upp mappen från Winless.
* Nu kommer Winless att hålla koll på din .less-fil. Varje gång du gör ändringar i filen och sparar så omvandlas .less-koden till css-kod.

**Variabler**

Som i annan programmering kan man ha variabler i .less.

I filen som du fick finns en variabel med värdet på en färg.

Den variabeln används sedan för att sätta färgen på figurerna.

Hade man inte använt variabel hade man fått klippa klistra för att ha samma färg.

Så nu kan vi ändra färgen på ett ställe istället för på varje figur.

@aColor: #FF4500;

#circle {

width: 150px;

height: 150px;

border-radius: 50%; /\* To create circle \*/

background-color: @aColor;

margin: 10px,

}

.square {

width: 150px;

height: 150px;

background-color: @aColor;

margin: 10px;

}

**HTML:**

<div id="circle"></div>

<div class="square"></div>

**Mixin**

Mixin innebär att man kan t.ex. använda klasser inuti andra klasser. Om man är van vi objektorientering kan det kännas bekant. Man ”ärver” från en klass till en annan.

.base {

border-radius: 50%; /\* To create circle \*/

background-color: @aColor;

margin: 10px;

}

.baseSmall {

.base;

width: 50px;

height: 50px;

}

.baseBig {

.base;

width: 250px;

height: 250px;

}

**HTML**

<div class="baseSmall"></div>

<div class="baseBig"></div>

**Räkna med CSS**

Man kan även använda matematik för att ändra värden.

Se följande kod som även använder de tidigare teknikerna:

@buttonBackground: #134d00;

@buttonColor: #ffff00;

@buttonPadding: 10px;

.button{

background: @buttonBackground;

color: @buttonColor;

padding: @buttonPadding;

display: inline-block;

border-radius: 4px;

}

.jumbo-button{

.button;

padding: @buttonPadding + 30; //Ökar paddingen med 30px.

}

.light-button{

.button;

background: @buttonBackground \* 1.8; //Ökar färgens värde. Ju högre ju ljusare. För mörkare minska.

}

**HTML**

<p><a href="#" class="button">Normal</a></p>

<p><a href="#" class="jumbo-button">Jumbo</a></p>

<p><a href="#" class="light-button">Light</a></p>

**Importera CSS**

Man kan även ha CSS I flera filer och importera dom till en fil.

Om vi har en fil som heter button.less med alla knappstilar och en som heter forms.less med alla formulär kan vi lägga till dom i en fil så här:

@import ”buttons”;

@import ”forms”;