

Delphi-indledning: Om properties og events

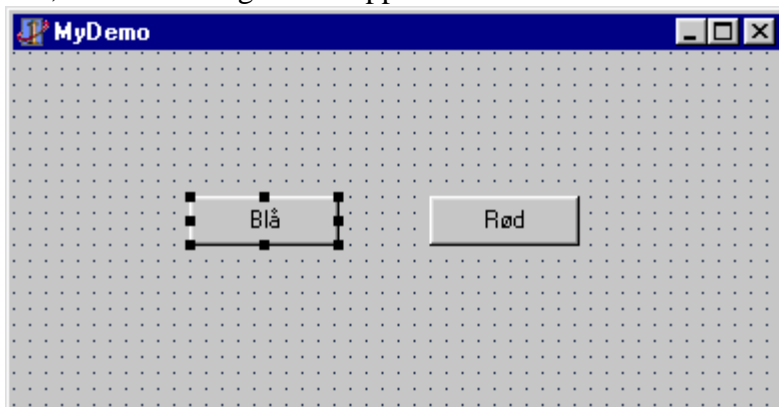
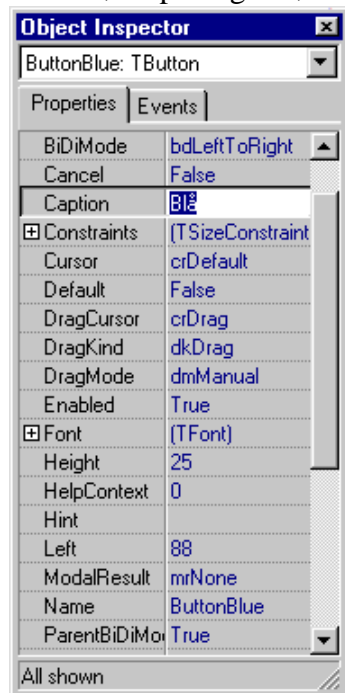
Om Delphi-komponenter gælder der, at

Enhver komponent har en række *properties* (egenskaber)

Til en komponent kan der være knyttet *events* (begivenheder eller hændelser)

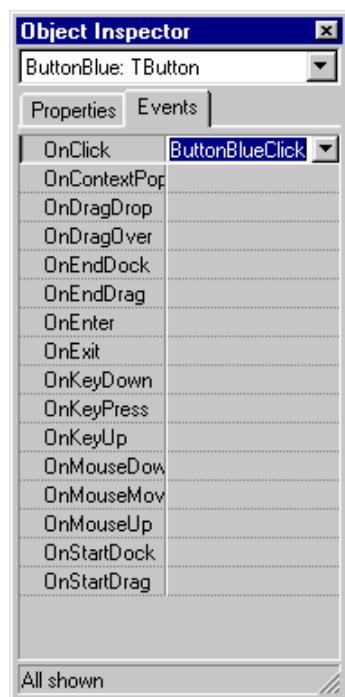
Komponentens properties defineres med Object Inspector, når man sætter komponenten på formen, men man kan også ændre på komponentens properties ved direkte programmering. Man kan f.eks. lade en eller flere events styre en komponents properties.

Vi ser først på bogens øvelse 2.1, hvor man bruger to knapper til at ændre formens farve.



Formen MyDemo (med navnet MyDemoForm) indeholder 2 Button-komponenter. På figuren er knappen med betegnelsen 'Blå' gjort til *det aktive objekt*. Knappens properties *Name* og *Caption* er ændret efter at knappen er sat på formen.

Name	Type	Caption	Evt. kommentar
MyDemoForm	TForm	MyDemo	
ButtonBlue	TButton	Blå	
ButtonRed	TButton	Red	



Til hver Button-komponent er der knyttet en *OnClick*-event. I programkoden for disse events står der, hvad der skal ske, fx når brugeren klikker på knappen med navnet *ButtonBlue*:

```
procedure TMyDemoForm.ButtonBlueClick(Sender: TObject);  
begin  
    MyDemoForm.Color := clBlue;  
end;
```

Bemærk notationen:

- MyDemoForm viser, at der er tale om en event på formen *MyDemoForm*, og ButtonBlueClick viser, at det er et klik på knappen *ButtonBlue*, der starter begivenheden.
- MyDemoForm.Color viser, at det er property'en *Color* på formen MyDemoForm der ændres, når man klikker på knappen...

Objektorienterede eller menustyrede programmer?

Programmer, der virker på denne måde, omtales ofte som *event-styrede*. Det er en metode, der er blevet meget udbredt de seneste år, i forbindelse med at *objektorienteret programmering* er blevet mere udbredt.

I objekt-orienteret programmering opbygger man programmerne omkring *objekter*, som man kan udføre handlinger på. Objekterne har en række *properties* (egenskaber), og handlingerne styres af *events* (begivenheder).

Også i brugen af programmerne kan man tale om objekt-orientering. Man vælger først det objekt, man vil arbejde med, dernæst hvilken handling man vil udføre. Object-Action.

Modsætningen til objektorienterede programmer er "gammeldags" menustyrede programmer. I menustyrede programmer vælger man sædvanligvis først, hvilken handling man vil udføre, dernæst hvilket objekt handlingen skal udføres på. Action-Object.

Som eksempel kan vi se på et kartoteksprogram, hvor man skal kunne tilføje kartotekskort, rette på eksisterende kartotekskort, eller slette kartotekskort.

Objektorienteret	Menustyret
1) Vælg (et eller flere) kartotekskort	1) Vælg menupunkt, f.eks.: "Rediger"
2) Vælg handling, f.eks. "Rediger"	2) Vælg (et eller flere) kartotekskort

Overgangen fra menustyrede programmer til objektorienterede programmer skete hastigt, især i forbindelse med overgange fra DOS til Windows. Windows er i sig selv objektorienteret.

Opgave: Find eksempler på objekter og handlinger i Windows-styresystem og -programmer.

Variable og værdier

De handlinger, der skal udføres, anbringes ofte i *procedurer* i programkoden. På forrige side havde vi et eksempel, hvor klik på en knap skulle medføre, at et objekts baggrundsfarve blev ændret:

```
procedure TMyDemoForm.ButtonBlueClick(Sender: TObject);  
begin  
  MyDemoForm.Color:=clBlue;  
end;
```

En procedure som denne "isolerer" så at sige handlingen. Dels så det er nemt at gennemskue hvad der foregår, dels så det er til at ændre eller tilføje yderligere handlinger, hvis man har lyst til det. Det kan jo være, at klikket på knappen ("ButtonBlueClick") skal medføre yderligere ændringer i objektet MyDemoForm.

Ændringen udføres inde i proceduren i *sætningen*

```
MyDemoForm.Color:=clBlue;
```

hvor *den variable* MyDemoForm.Color *får tildelt værdien* clBlue.

Bemærk notationen, hvor *værdi-tildelingen* sker med symbolet := (kolon lig med), der læses som "sættes til at være".

Tildelings-sætning	variabel := værdi
--------------------	-------------------