Bindningar

- dvs att binda identifierare till olika enheter
- påverkar programs flexibilitet
- begränsar omfattningen av viss typ av förändringar

Centrala begrepp

- bindningar
- räckvidd (scope)
- synlighet (visibility)
- deklarationer
- block

Bindningar och omgivningar

- användning av identifierare måste alltid ses i sitt sammanhang
- ett *sammanhang* eller en *omgivning* är en mängd olika bindningar som hör ihop

Bindningsbara värden

(bindables)

- I ML är det:
  - primitiva värden, sammansatta värden, funktionsabstraktioner, referenser till variabler (i värdedefinitioner)
  - typer (i typ- och "datatype"-deklarationer
  - undantag (i exceptions-deklarationer)

Räckvidd

(scope)

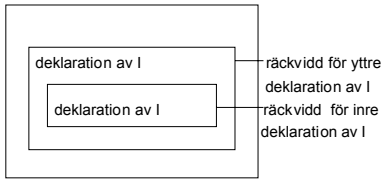
En deklarations räckvidd är den del av programtexten där den är verksam

Blockstruktur

Programfras som avgränsar räckvidden för en bindning

- monolitisk blockstruktur
- platt blockstruktur
- nästad blockstruktur

Räckvidd och synbarhet



Statisk och dynamisk bindning

- Hör samman med typningen

Statisk bindning:

- blocket evalueras i den omgivning som ges vid bindningstillfället
- avgörs vid kompilering

Dynamisk bindning

- blocket evalueras i den omgivning som ges vid användningstillfället
- avgörs vid exekvering

Deklaration

En deklaration är en programfras som producerar bindningar

- definitioner
- typdeklarationer
- variabeldeklarationer
- samtidig deklaration
- sekvensiell deklaration
- rekursiv deklaration

Definitioner

En definition är en enkel deklaration vars enda verkan är att producera en bindning till exempelvis konstant, typ, procedur, funktion

Typdeklaration

- syftet med en typ*definition* är att binda en identifierare till en existerande typ (strukturell ekvivalens)
- en ny-typ-*deklaration* skapar en ny, distinkt typ (namnekvivalens)

Variabeldeklaration

Variabeldefinition binder en identifierare till en existerande variabel, ovanligt men finns i ADA (renaming)
Variabel*deklaration* binder identifierare till ny variabel

Samtidig deklaration

I ML:

```
val pi = 3.14159
and sin = fn (x: real) => ...
and cos = fn (x: real) => ...
```

Men inte

```
and tan = fn (x: real)
    => sin (x) / cos (x)
```

Sekventiell deklaration

- Den vanligaste formen av deklaration
- Kan utnyttja tidigare gjorda bindningar

Rekursiv deklaration

- Kännetecknas av att den använder bindningar som den själv producerar
- Mest användbar i typ-, procedur- och funktionsdefinitioner
- Automatiskt rekursiv eller specificerat rekursiv

Deklarationers räckvidd

- Enkla deklarationer: från deklarationens slut till blockets slut
- Samtidiga deklarationer: från (hela) deklarationens slut till blockets slut
- Sekventiella deklarationer: den första är tillgänglig från den andra osv

Block

- Ett block är en programfras som begränsar räckvidden för de deklarationer den innehåller

  Några tänkbara blocktyper:
  - blockkommandon
  - blockuttryck

Blockkommandon

- deklarationer lokala för kommandon

```
D begin C end
```

Blockuttryck

- deklarationer lokala för uttryck

```
let D in E end
```

Kvalifikationsprincipen

Det är möjligt att inkludera ett block i varje syntaktisk kategori, förutsatt att fraserna i den klassen specificerar någon sorts beräkning
Exempel: En *blockdeklaration* är ett block innehållande lokala deklarationer, vilka endast används för att utföra blockdeklarationer