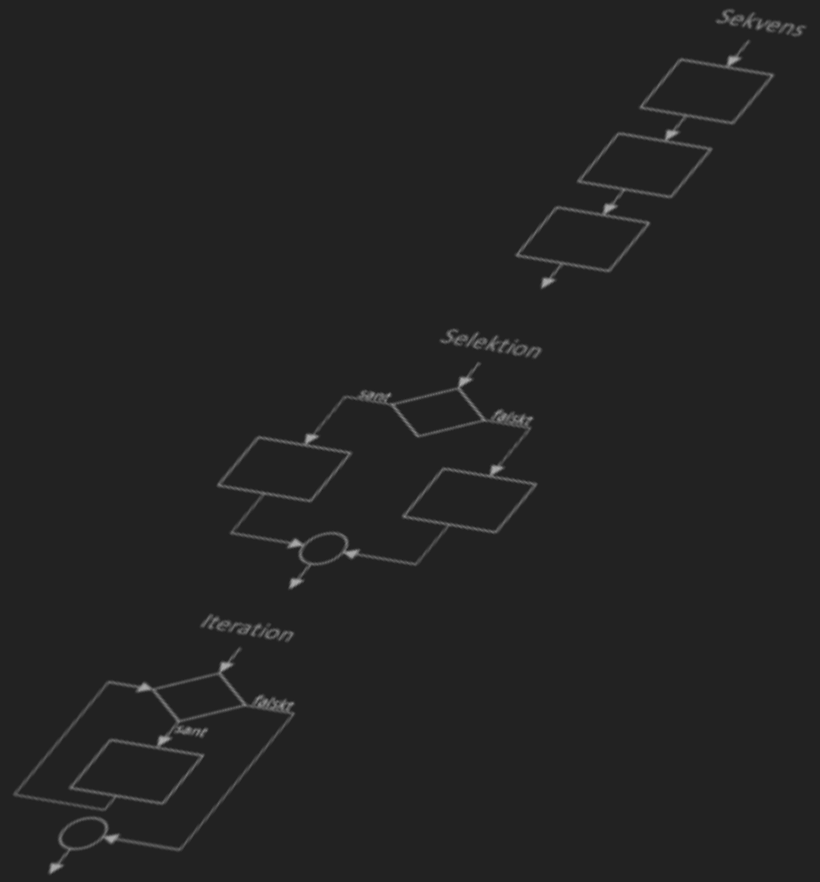
Styrstrukturer och kommentarer

Kontrollstrukturer

- ✓ Kontrollstrukturer är grundläggande verktyg vid programmering.
- ✓ Sekvens
  - En följd satser som måste utföras efter varandra.
- ✓ Selektion
  - Ett sätt att välja mellan att exekvera olika satser.
  - `if`-satsen, `switch`-satsen, villkorsoperatorn (?:)
- ✓ Iteration
  - Ett sätt att upprepa en sats ingen eller flera gånger.
  - `while`-satsen, `do...while`-satsen, `for`-satsen.
- ✓ Rekursion
  - En funktion som anropar sig själv. (Mer om detta längre fram i kursen.)



Sekvens

- En sekvens är en följd av instruktioner.

```
Instruktion 1
Instruktion 2
Instruktion 3
Instruktion 4
```

- Den enklast möjliga kontrollstrukturen, men då en instruktion kan innehålla andra kontrollstrukturer kan det bli komplicerat.

Selektion

- Enkelt val (`if`-satsen)

```
OM villkor uppfyllt
  Instruktion(er)
```

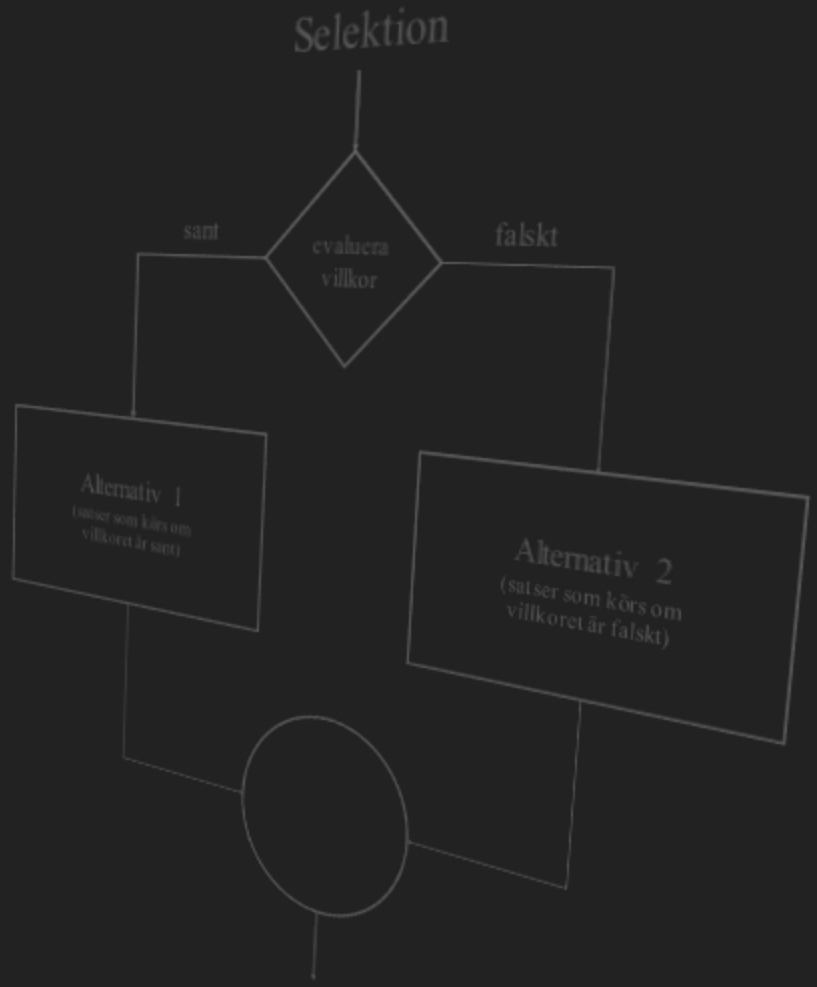
- Tvåvägsval (`if...else`-satsen)

```
OM villkor uppfyllt
  Alternativ 1
ANNARS
  Alternativ 2
```

- Flervägsval (`if...elseif...else`-satsen, `switch`-satsen)

```
OM villkor uppfyllt
  Alternativ 1
ANNARS OM villkor uppfyllt
  Alternativ 2
ANNARS OM villkor uppfyllt
  Alternativ 3
ANNARS
 Alternativ 4
```

```
VÄLJ fall ur
  FALL 1: Alternativ 1
  FALL 2: Alternativ 2
        .
        .
        .
  ANNARS: Alternativ n
```



`if`-satsen

```
if (villkorsuttryck) {
    sats(er) // Omslut alltid satsen/satserna mellan klammerparenteser.
}            // (Behövs inte om det bara är en sats, men gör det alltid ändå.)
```

- ✓ Om villkorsuttrycket är `true` utförs en eller flera satser omslutna mellan klammerparenteserna, `if`-satsens kropp.
- ✓ Villkorsuttrycket är ett uttryck som resulterar i något som JavaScript betraktar som `true` eller `false`.

```
let number = 42
if (number === 0) {
  console.log('Talet är lika med 0.')
}
```

`if...else`-satsen

```
if (villkorsuttryck) {
    sats(er)1
} else {
 sats(er)2
}
```

- ✓ Endast ett av alternativen kommer att utföras.
- ✓ Är villkorsuttrycket...
  - ...`true` utförs en eller flera satser som följer direkt efter villkorsuttrycket.
  - ...`false` utförs en eller flera satser som följer direkt efter `else`.

```
let number = 42
if (number % 2 === 0) {
  console.log('Talet är jämt.')
} else {
  console.log('Talet är udda.')
}
```

if...else if...else-satsen

```
if (villkorsuttryck) {
   sats(er) 1
} else if (villkorsuttryck2) {
 sats(er) 2
} else if (villkorsuttryck3) {
 sats(er) 3
} else
 sats(er) 4
}
```

- ✓ Villkorsuttryckens värde bestäms till och med det första som blir `true`.
- ✓ Avslutande `else` kan uteslutas.

```
let number = 42
if (number < 0) {
  console.log('Talet är mindre än 0.')
} else if (number < 50) {
  console.log('Talet är större än 0 men mindre än 50.')
} else {
  console.log('Talet är större än, eller lika med, 50.')
}
// OUTPUT: Talet är större än 0 men mindre än 50.
```

`switch`-satsen

- Kan vara ett alternativ *(pun not intended!)* istället för en `if...else if...else`-sats.

```
let number = 4
switch (number) {
  case 1:
  case 2:
  case 3:
  case 4:
  case 5:
  case 6:
    console.log('För litet!')
    break
  case 7:
    console.log('Rätt gissat')
    break
  case 8:
  case 9:
  case 10:
    console.log('För stort!')
    break
  default:
    console.log(number + ' är inte i det slutna intervallet mellan 1 och 10.')
    break
}
// OUTPUT:  För litet!
```

Villkorsoperatorn

- En vanlig `if...else`-sats kan (ibland lämpligen) uttryckas...

```
const number = 42
let output = 'Talet är '
if (number % 2 === 0) {
  output += 'jämt.'
} else {
  output += 'udda.'
}
console.log(output)

// OUTPUT: Talet är jämt.
```

- ...med hjälp av villkorsoperatorn `?:`.

```
const number = 42
let output = 'Talet är '
output += number % 2 === 0 ? 'jämt.' : 'udda.'
console.log(output)

// OUTPUT: Talet är jämt.
```

Iteration

- ✓ Förtestad upprepning (`while`-satsen)
  - Loopen körs ingen, en eller flera gånger.

```
SÅ LÄNGE villkor uppfyllt
    Instruktion(er)
```

- ✓ Eftertestad upprepning (`do...while`-satsen)
  - Loopen körs minst en gång.

```
REPETERA
    Instruktion(er)
SÅ LÄNGE villkor uppfyllt
```

- ✓ Bestämd upprepning (`for-satsen`)
  - Loopen körs ingen, en eller flera gånger.

```
initiera räknare
SÅ LÄNGE räknarvillkor uppfyllt
    Instruktion(er)
uppdatera räknare
```

while-satsen

```
while (villkorsuttryck) {
  sats(er) // Omslut alltid satsen/satserna mellan klammerparenteser.
}          // (Behövs inte om det bara är en sats, men gör det alltid ändå.)
```

- Så länge som villkorsuttrycket utvärderas till `true` utförs satserna i loopens kropp.

```
let sum = 0
while (sum < 500) {
  sum = sum + 42
}
console.log(sum)

// OUTPUT: 504
```

do..while-satsen

```
do {
  sats(er) // Omslut alltid satsen/satserna mellan klammerparenteser.
} while (villkorsuttryck)
```

- Satserna i loopens kropp körs minst en gång och så länge som villkorsuttrycket utvärderas till `true` fortsätter satserna i loopens kropp att upprepas.

```
let i = 10
do {
  console.log(i + ' ')
  i++
} while (i < 10)

// OUTPUT: 10
```

Satserna i loopens kropp utförs bara en gång eftersom villkorsuttrycket är `false`!

for-satsen

```
for (initiering räknare; villkorsuttryck; uppdatering räknare) {
  sats(er) // Omslut alltid satsen/satserna mellan klammerparenteser.
}
```

- ✓ En `for-sats` är ett annat sätt att uttrycka en `while`-sats.
  1. initiering räknare.
  2. villkorsuttryck
  3. sats(er)
  4. uppdatering räknare (gå till punkt 2)

```
let output = ''

for (let i = 0; i < 10; i++) {
  output += i + ' '
}

console.log(output)

// OUTPUT: 0 1 2 3 4 5 6 7 8 9
```

Avbryta en loop

- `break`-satsen avbryter en loop.

```
let output = ''

for (let i = 0; i < 10; i++) {
  if (i === 4) {
    break
  }
  output += i + ' '
}

console.log(output)

// OUTPUT: 0 1 2 3
```

Gå till nästa iteration av en loop

- `continue`-satsen avslutar aktuell iteration av loopen och påbörjar nästa iteration av loopen.
  - Anses av vissa att vara "*bad practice*" att använda. Kod kan i regel skrivas så att `continue`-satsen inte behöver användas.

```
let output = ''

for (let i = 0; i < 10; i++) {
  if (i % 3 === 0) { // 3, 6 och 9 jämt delbart med 3
    continue
  }
  output += `${i} ` // en "template literal", ger samma resultat som uttrycket output = i + ' '
}

console.log(output)

// OUTPUT: 1 2 4 5 7 8
```

Kommentarer

- ✓ Kod säger inte allt. För att beskriva vad kod gör kan kommentarer användas.
- ✓ Kommentarer är VIKTIGT att skriva.
  - Ska lämpligen stå på helt egna rader med en tom rad raden ovan.
- ✓ Det finns olika typer av kommentarer.

```
let product = 1

// Radkommentarer går till slutet av raden.
for (let i = 3; i < 100; i += 2) {
  product *= i
}

/* Blockkommentarer där allt mellan start och slut
   betraktas som en kommentar. */
console.log(product)

/**
 * (Dokumentationskommentarer (JSDOC).)
 * Returns a string where all...
 *
 * @param {string} data - The string being...
 * @returns {string} A new string with...
 */
const foo = function (data) {
  // do something with data to create some return value...
  return someReturnValue
}
```