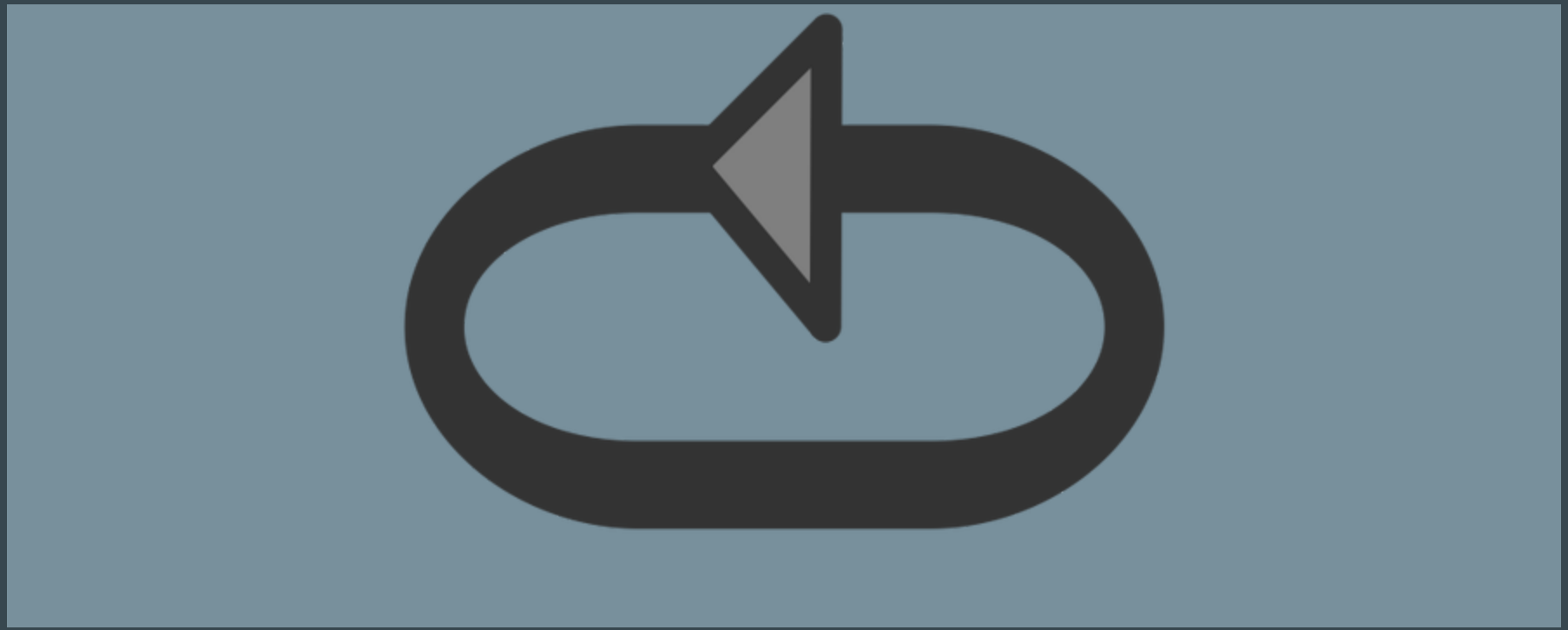
Programmering - Lektion 4

PC 230915

Från förra veckan



for-loopen

```
for (var x = 100; x<=700; x += 100) {
    circle(x, 200, 40, “green”);
}
```

När jag vet hur många gånger jag ska upprepa är det oftast en for-loop jag använder.

for-loopen

```
for (initalisering; test; storlek på steg) {
  // Kod som körs i loopen
}
```

I JavaScript är for-loopen ganska löst definierad men om vi använder den på det här viset kommer det fungera.

Initialisering kan vi tänka på som startvärdet, vi deklarerar vanligtvis en variabel här.

Test kollas inför varje loop och måste vara ett villkor.

Slutligen kan vi ange hur mycket vi vill att variablen vi deklarerade i initialisering ska öka.

while loopar

```
let x = 50;
while (x <= 700){
  circle(x, 100, 20, "green");
  x += 70;
}
```

Om jag inte vet hur många gånger gånger jag behöver upprepa en loop så finns while loopen.

Det här exemplet fungerar egentligen lika bra med en for-loop.

while loopen

```
while (condition){

  //kod som körs om condition är sant

}
```

Om jag inte vet hur många gånger gånger jag behöver upprepa en loop så finns while loopen.

while loopen

```
let x = 100;

let radie = 20

while (radie <= 500){

   circle(x, 100, radie, "orange");

   x += 100;

  radie=random(600);

}
```

Här är ett exempel där vi ritar slumpartade cirklar tills radien blir större än 500.

Hur många Hej?

```
let i = 100;

while (i <= 200) {

    text(10, i, 12, "Hej", “red”);

    i += 20

}
```

Hur många “Hej” kommer skrivas ut på skärmen?

A. 6
B. 100
C. 200

Skillnaden mellan for och while?

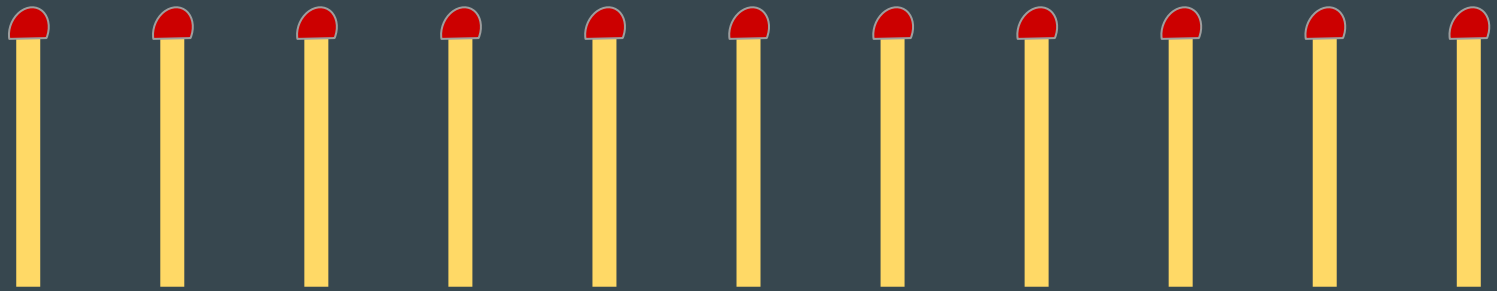
- I JavaScript är skillnaden inte så stor, en loop kan antingen göras med hjälp av for eller while.
  - For ska användas när antalet upprepningar är känt
  - While när vi inte på förväg vet hur många upprepningar det tar tills ett villkor är uppfyllt.
- simple.js gör att vi inte behöver använda loopar i samma utsträckning, funktionen update() är i princip en loop.
- Det finns andra loop konstruktioner, JavaScript har även en do-while loop.
- Se upp för oändliga loopar - Spara din kod innan du kör det!

  Felaktig kod kan bli dyrt.

Spelet Nim

Ett klassiskt spel med tändstickor eller andra små objekt. Här är en förenklad variant.

I vårt exempel börjar vi med 12 tändstickor. Du börjar och får ta 1, 2 eller 3 stickor. Den som tar sista sticka vinner.



Spela med en kompis och se om ni kan hitta en strategi...

Om du börjar kommer du alltid att förlora...

- Den som tar fjärde stickan kommer förlora
- eller den som tar åttonde sticka kommer förlora..
- Vi ska se till att summan av stickorna som tas i en omgång blir 4
- Spelare 1 börjar
  - Om spelare 1 tar 1 sticka tar du 3 stickor
  - Om spelare 1 tar 2 stickor så tar du 2 stickor
  - Om spelare 1 tar 3 stickor så tar du 1 sticka.

Textbaserat Nim

1. Vilka variabler behöver vi?
2. Hur ska spelaren mata in sitt val?
3. Vilken typ av loop är lämplig?
4. Hur ska vi presentera resultatet?

1. Totala antalet stickor, spelarens val och datorns val
2. Just nu använder vi en enkel prompt
3. Vi vet inte antalet upprepningar; while
4. Just nu med en alert box.

Enkelt text Nim

```
let stickor = 12;
let user = 0;
let computer = 0;

while (stickor > 0){
  user = prompt("Det finns " + stickor + " stickor. Hur många stickor tar du (1-3)?");

  stickor -= user;
  computer = 4 - user;
  alert("Jag tar " + computer + " stickor");
  stickor -= computer;
}
alert("Jag vann!");
```

Datatypen Sträng

I JavaScript finns det en datatyp som heter string. På svenska säger vi sträng.

Det är en följd av bokstäver eller snarare tecken.

Strängar omges av antingen dubbla eller enkla citationstecken.

I övningarna från förra veckan fanns några uppgifter där ni skulle manipulera strängar.

Låt användaren mata in en textsträng. Skriv program som matar ut :

- Första bokstaven
- sista bokstaven
- namnet skrivet baklänges

Exempel på strängar

```
namn = "Peter";

color = 'BlanchedAlmond';

bgColor ="DarkSlateBlue";

fill(bgColor);

greeting = "Hej " + namn;

text(10,100, 50, greeting, color);
```

Strängar omges av dubbla eller enkla fnuttar.

Om man vill slå ihop två strängar använder man + tecken. Att slå ihop strängar kallas för *konkatenering*.

Sträng som ett objekt

```
namn = "Peter";

greeting = "Hej " + namn;

text(10,100,50, greeting.length,
color);
```

Strängen är ett objekt med flera egenskaper och värden knuten till sig. Vi kan ta reda på strängens längd och individuella bokstäver (egentligen tecken) i strängen.

Första bokstaven

	P	e	t	e	r
index	0	1	2	3	4

```
namn = "Peter";

b = namn.charAt(0);

text(10,100,50, b, color);
```

För att hämta en enskild bokstav så använder vi charAt(index). Strängar indexeras från noll, så om jag vill ha första bokstaven så behöver jag titta i index noll.

Sista bokstaven

	P	e	t	e	r
index	0	1	2	3	4

```
namn = "Peter";

b = namn.charAt(name.length-1);

text(10,100,50, b, color);
```

För att hämta en enskild bokstav så använder vi charAt(index). Strängar indexeras från noll så jag måste titta i index `length - 1`

Vända på en sträng

	Ursprunglig sträng				
	P	e	t	e	r
index	0	1	2	3	4
	Baklänges sträng				
	r	e	t	e	P
index	0	1	2	3	4

Läs strängen bokstav för bokstav baklänges

Bokstaven i index 4 ska läggas in i index 0 i den nya strängen.

Tips det går att göra for-satser som räknar nedåt.

Vända på en sträng

	Ursprunglig sträng				
	P	e	t	e	r
index	0	1	2	3	4
	Baklänges sträng				
	r	e	t	e	P
index	0	1	2	3	4

```
namn = "Peter";

  back = ""  // Tom sträng

  //Vänd på en sträng med hjälp av en
  //for-sats som räknar nedåt

  for (var i =namn.length-1 ; i>=0; i--){

    back = back + namn.charAt(i);

  }

  text(10,100, 50, namn, "black");

  text(10,170, 50, back, "red");
```

Nästa vecka - funktioner

Om jag vill vända på en sträng flera gånger kan jag skapa en funktion.

```
namn ="PeterC";

function bakOchFram(inString){
  var back =""
  for (var i =namn.length ; i>=0; i--){
    back = back + namn.charAt(i);
  }
  return back;
  }

bak = bakOchFram(namn);
alert(bak);
```