

JavaScript

Först lite repetition

```
var num = 10;

while( num >= 0){
    if ( num % 2 == 0){
        console.log(num);
    }
}
```


While loopen sköter
loopandet

If-satsen kollar
villkoret för jämna tal

for loop

```
var sum = 0;
```

Deklarera variabeln
utanför loopen



```
for(var i = 1; i <=10; i++){  
    sum = sum + i;  
}
```

Addera till variabeln
Innanför loopen



String concatenation

Hur vi sätter ihop värden och strängar

```
console.log("Mitt namn är " + name);
```

```
var a = 100;
```

```
var b = "" + a;
```

```
var b = a.toString();
```

```
var b = String(a);
```

Era bästa vänner



stackoverflow



Mjau Machine

```
console.log("mjau ".repeat(4));
```



Inbyggd funktion



Funktioner

Effektiv kodning

- **DRY** - Don't Repeat Yourself
- **DO ONE THING**
- **Refactor:**

”Code **refactoring** is the process of restructuring existing computer code—changing the factoring—without changing its external behavior. ”

Funktion

- En funktion kapslar in kod:
Encapsulation
- Låter oss återanvända kod
- Gör vi något ofta:
 - **Gör en funktion av det!**

Funktioner tar det ett steg längre

Alltid nyckelordet `function`

Namn på funktionen

```
function myFunction(){  
    //Do stuff  
}
```

kodblock

Ett exempel

```
function speak(){  
    console.log('I can speak!!');  
}
```

```
speak();
```

```
//I can speak!!
```

Koden körs inte
förrän vi kallar på
den

Kallar på den genom
namnet samt
paranteser

Scope

```
function speak(){  
    var name = 'Jesper';  
    console.log('I can speak!!');  
}
```



Local scope

A dark gray rectangular box labeled "Local scope" is connected to the function definition code above it by a black curly bracket on the left side.

```
console.log(name);
```



Global scope

A pink rectangular box labeled "Global scope" is connected to the `console.log(name);` statement above it by a pink arrow pointing from the box to the variable `name`.

Återanvänd kod

```
Speak();  
Speak();  
Speak();
```

```
//I can speak!!  
//I can speak!!  
//I can speak!!
```

Den här funktionen
är dock inte så
givande ☹

Parametrar

- Vi kan skicka med värden in i funktioner
- Dessa värden kallas **parametrar**

```
function sayMyName(name){  
    console.log(name);  
}
```



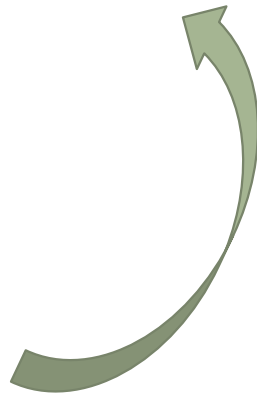
Parameter

En variabel som kan användas inom funktionen

Parameter exempel

```
function sayMyName(name){  
    console.log(name);  
}
```


```
sayMyName('Jesper');
```



Argumentet 'Jesper' skickas in i funktionen där det kan användas

Flera parametrar

Separerade med komma



```
function sayMyName(firstName, lastName){  
    console.log(firstName + lastName);  
}
```


return

- Oftast ska en funktion **returnera** ett värde
- Nyckelordet **return**
- När vi kallar på en funktion får vi tillbaka det värde som returneras inuti funktionen

return

```
var sum = 0;
```

```
for ( var i = 0; i <= 10; i++){
```

```
    sum = sum + i;
```

```
}
```

```
console.log(sum);
```

return exempel

```
function sayMyName( name ){  
    return name;  
}
```

```
sayMyName( 'Jesper' );  
//Jesper  
sayMyName( 'Anti-Jesper' );  
//Anti-Jesper
```

Lagra returvärde

```
function sayMyName(name){  
    return "Mitt namn är " + name;  
}  
var name = sayMyName(name);
```

Visa returnvärde

```
function sayMyName(name){  
    return "Mitt namn är:" + name;  
}
```

```
console.log(sayMyName('Jesper'));
```

Just nu

- Ta rast!
 - Koda!
 - Ta rast!
 - Koda!
-
- Efter lunch: lösningsförslag och fördjupning