

Repetition av JS

...eller: "Det här hade varit bra att kunna innan Webbtjänster-kursen, Johan!"

Dagens föreläsning

- Bakgrund
- Våra verktyg
- Grunderna i Javascript
- Lite mer avancerad Javascript

Bakgrund

Vad är Javascript och varför ska vi bry oss?

Språket Javascript

- Ursprungligen avsett för att göra webbsidor roligare
 - Bra stöd för att manipulera DOM
 - Bra stöd i de flesta webbläsare
- Körs numera lite över allt
- Har numera ett bra stöd av tredjepartsbibliotek

Språket Javascript

- Syntaxen liknar C, Java, etc
- Löst typat – inga explicita variabeltyper
- Multiparadigm:
 - Händelsedrivet
 - Funktionellt
 - Prototypbaserat (ungefär som objektorienterat)
 - Imperativt

Exempel på användning

- Dynamiska webbsidor
- Serverapplikationer med Node.JS
- Mobilapplikationer med Cordova
- Spelutveckling med Unity
- Robotstyrning med Nodebots

Lite historik



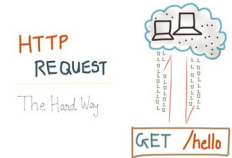
1995: Netscape



1996: JScript



1997: ECMAScript



2005: AJAX



2006: JQuery



2008: V8-motorn



2010: Node.JS



2015: ES 6

Våra verktyg

Våra verktyg

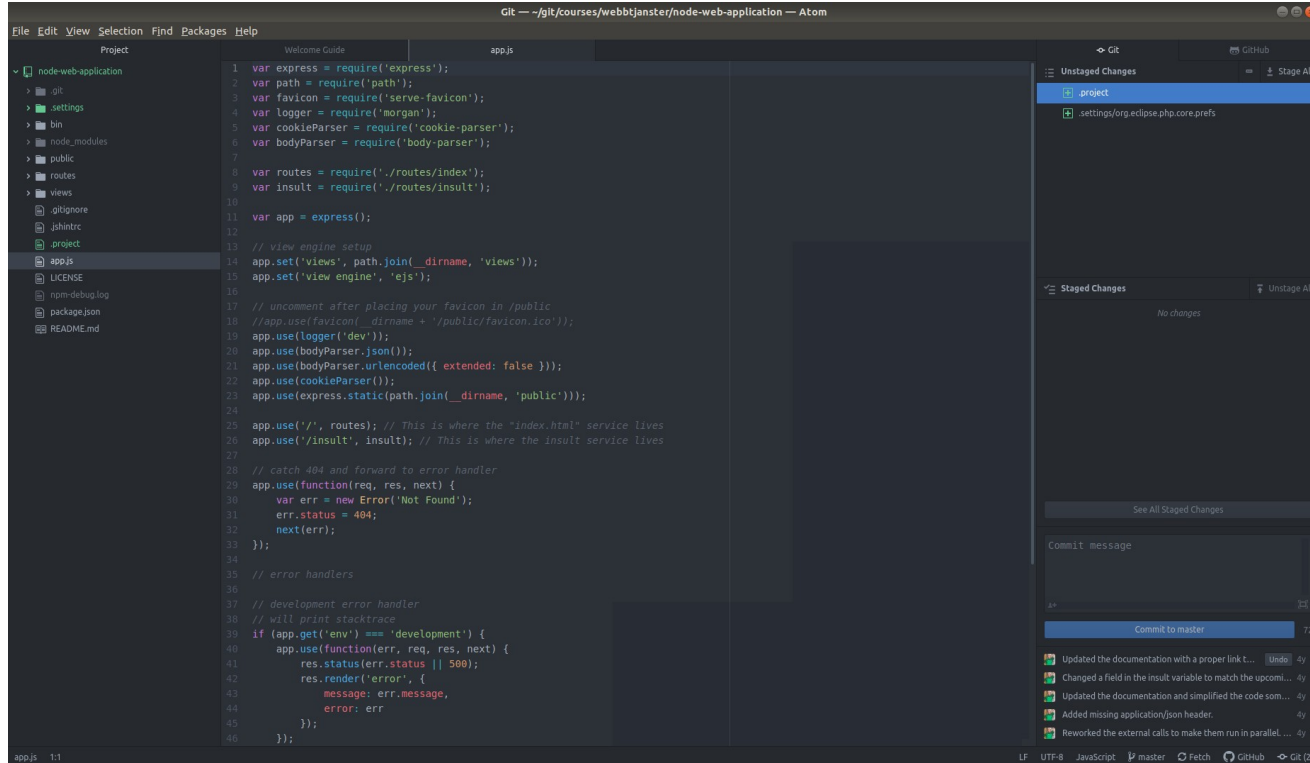
Textredigerare: Atom

Webbläsare: Chromium med Chrome Developer Tools

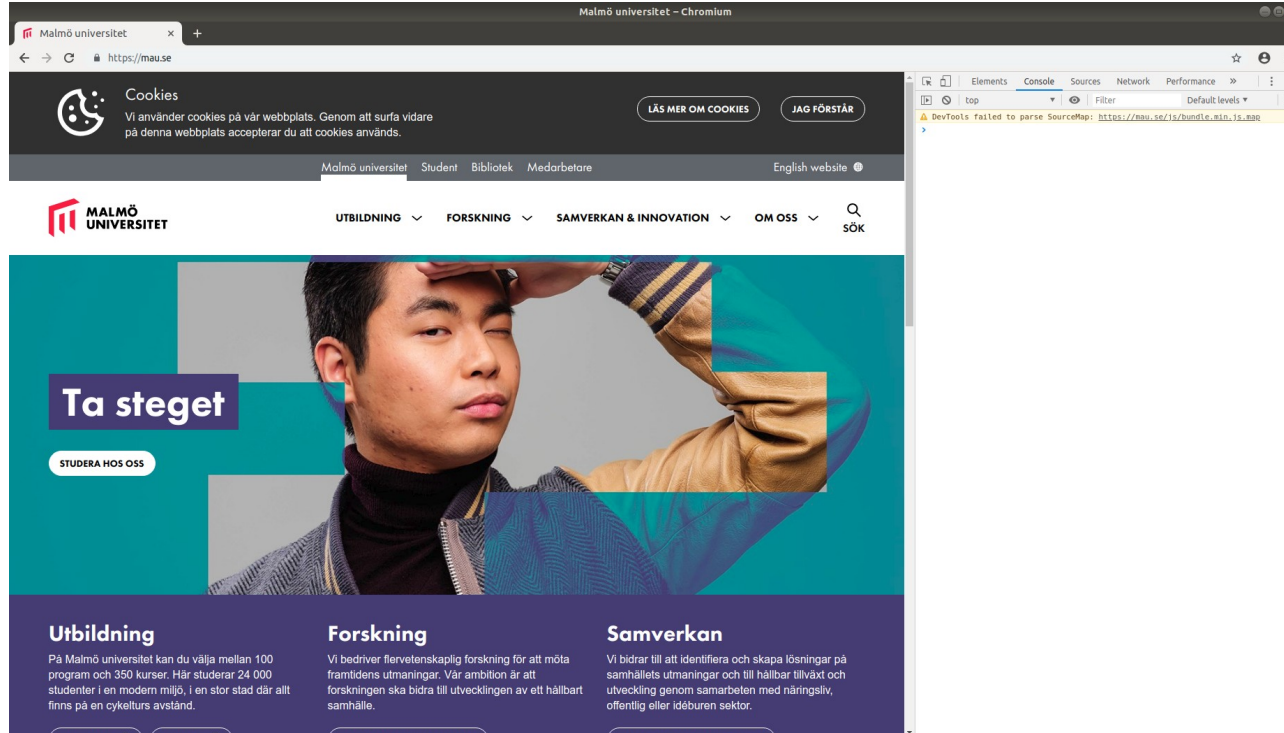
Onlineverktyg: JSFiddle

Terminal: cmd, Terminal, etc

Textredigeraren Atom



Webbläsaren Chromium



Onlineverktøyet JSFiddle

Simple function example

+

https://jsfiddle.net/maclopp/vevdoao/

Run

Save

Fork

Tidy

Collaborate

Embed


Update

New and updated JS/CSS/Inters

Settings

Sign in

Author



Mindy McAdams

Gainesville, FL

Fiddle meta

Resources [URL](#) [cdjs](#)

Async requests

Other (links, license)

HTML

```

1 <p>Type a number: <input type="text" id="f" size="4"></p>
2
3 <p>Type a different number: <input type="text" id="s" size="4"></p>
4
5 <button id="b" class="myButton">Get the larger</button>
6
7 <p>The larger of the two numbers is:
8 <span id="write"></span></p>

```

CSS

```

1 {
2   font-family: 'Consolas', monospace;
3   font-size: 1.1em;
4   padding: 0 5px;
5   text-align: right;
6 }
7
8 /* Inline button CSS from http://www.besccsbuttongenerator.com/ */
9 .myButton {
10   -moz-box-shadow: 0px 1px 0px 0px #0f77fa;
11   -webkit-box-shadow: 0px 1px 0px 0px #0f77fa;
12   box-shadow: 0px 1px 0px 0px #0f77fa;
13   background: -webkit-gradient(linear, left top, left bottom, color-stop(0.85, #33bdef), color-stop(1, #019ad2));
14   background: -moz-linear-gradient(top, #33bdef 5%, #019ad2 100%);
15   background: -webkit-linear-gradient(top, #33bdef 5%, #019ad2 100%);
16   background: -ms-linear-gradient(top, #33bdef 5%, #019ad2 100%);
17   background: linear-gradient(to top, #33bdef 5%, #019ad2 100%);
18   background: linear-gradient(to bottom, #33bdef 5%, #019ad2 100%);
19   filter: progid:DXImageTransform.Microsoft.Gradient(startColorstr="#33bdef", endColorstr="#019ad2", GradientType=0);
20   background-color: #33bdef;
21   -ms-border-radius: 5px;
22   -webkit-border-radius: 5px;
23   border-radius: 5px;


```

JavaScript + No-Library (pure JS)

```

1 // Get the button and when it responds is a click
2 var theButton = document.getElementById("b");
3 theButton.onclick = feedTheButton;
4
5 // single function compares two numbers, returns the larger one
6 function greatestOfTwo( first, second ) {
7   if ( first > second ) {
8     return first;
9   } else {
10    return second;
11  }
12 }
13
14 // this function runs each time the button is clicked
15 // the simple function is called within this one
16 function feedTheButton() {
17   // get the two numbers from the text input fields
18   // parseInt() changes string to number
19   var box1 = parseInt( document.getElementById("f").value );
20   var box2 = parseInt( document.getElementById("s").value );
21   // run the function above this one and store what is returned in result
22   var result = greatestOfTwo( box1, box2 );
23   // write the result into the HTML
24   var place = document.getElementById("write");
25   place.innerHTML = result;

```



Project tracking, teamwork & client reporting like you've never seen before.

ads via Carbon

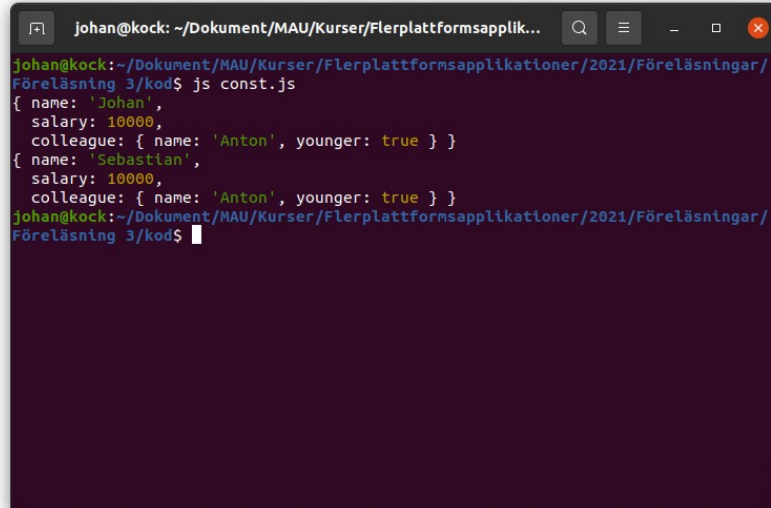
Type a number:

Type a different number:

Get the Larger

The larger of the two numbers is:

Terminalen



```
johan@kock: ~/Dokument/MAU/Kurser/Flerplattformssapplik...  
johan@kock:~/Dokument/MAU/Kurser/Flerplattformssapplikationer/2021/Föreläsningar/  
Föreläsning 3/kod$ js const.js  
{ name: 'Johan',  
  salary: 10000,  
  colleague: { name: 'Anton', younger: true } }  
{ name: 'Sebastian',  
  salary: 10000,  
  colleague: { name: 'Anton', younger: true } }  
johan@kock:~/Dokument/MAU/Kurser/Flerplattformssapplikationer/2021/Föreläsningar/  
Föreläsning 3/kod$
```

Grunderna i Javascript

Satser

Instruktioner i Javascript kallas *statements* eller *satser*.

Varje sats *kan* avslutas med ett semikolon (eller radbrytning)

```
console.log("hello, world!");
```

Flera satser kan placeras i ett *block*. Dessa avgränsas med *måsvingar* eller *kryllparenteser*

```
{  
  console.log("hello");  
  console.log("world");  
}
```

Tilldelning

Variabler får sina värden med hjälp av ett = -tecken

```
a = 1; // Tilldelning  
b = true; // Tilldelning
```


If- och switch-satser

If-satser används för att hantera villkor:

```
if (a == 2) {  
    // Do something  
} else if (a == 4) {  
    // Do something  
} else {  
    // Do something else  
}
```

Switch-satser fungerar på ungefär samma sätt:

```
switch (a) {  
    case 2:  
        // Do something  
        break;  
    case 4:  
        // Do something  
        break;  
    default:  
        // Do something else  
}
```

Loopar

Två typer av loopar: while och for

```
while (true) {  
    // Do something  
}
```

```
for (var i = 0; i < 10; i++) {  
    // Do something  
}
```

Operationer

// Aritmetik

```
a = 1 + 2; // Addition
a = 2 - 1; // Subtraktion
a = 2 * 2; // Multiplikation
a = 2 / 2; // Division
a = 5 % 2; // Resten
a++; // a = a + 1
a--; // a = a - 1
a += 2; // a = a + 2
a -= 2; // a = a - 2
b = !b; // B är nu falskt
```

// Jämförelser

```
a == 2; // Är a lika med två?
a === b; // Är a precis samma
           objekt som b?
a != 2; // Är a skilt från två?
a < 2; // Är a mindre än två?
a > 2; // Är a större än två?
a && b; // Är a och b båda sanna?
a && !b; // Är a sant och b falskt?
a || b; // Är a eller b sanna?
```

Variabler

```
var a; // Initaliserad, inget värde
var b = 1; // Initaliserad, har ett värde
let c = true; // Initaliserad, har ett värde
const d = "hello"; // Initaliserad, har ett värde
console.log(a); // Skriver ut "undefined"
console.log(b); // Skriver ut "1"
console.log(c); // Skriver ut "true"
console.log(d); // Skriver ut "hello"

b = 5; // Nytt värde!
console.log(b); // Skriver ut "5"

d = "goodbye"; // Här går det fel!
```

Variabler

- Löst typade (sen nästa slide)
- Deklareras på tre sätt:
 - `var` – äldre sätt, synligt och kan omdefinieras det aktuella scopet
 - `let` – nyare sätt, synligt, men kan inte omdefinieras i det aktuella scopet, men kan omdefinieras i ett block
 - `const` – nyare sätt, konstant och synligt i aktuellt scope

Datatyper

Javascript har fem datatyper (ish...)

- String
- Number
- Boolean
- Object
- Array

Strängar - String

En lista av tecken:

```
var a = "hello, world!";
```

Längden av en sträng:

```
console.log(a.length); // Skriver ut "13"
```

Hämta ut ett tecken ur en sträng:

```
console.log(a[2]); // Skriver ut "l"
```

Tal - Number

Vanliga hel- och flyttal:

```
var a = 1; // Heltal
```

```
var b = 1.5; // Flyttal
```

Och en specialare: NaN!

```
var c = NaN; // Inte ett nummer, men ändå ett nummer.
```


Sant/falskt - Boolean

Representerar värdena sant och falskt:

```
var iAmWeazel = true;
```

```
var iRBaboon = false;
```

Objekt - Object

Används för att hålla koll på många värden. Kan innehålla *vad som helst!*

```
var teacher = {  
  name: "Johan",  
  salary: 10000,  
  colleague: {  
    name: "Anton",  
    younger: true  
  }  
};
```

Vektorer/listor - Array

En typ av objekt som håller koll på värden i en specifik ordning:

```
var a = new Array();  
a[0] = 1;  
a[1] = "two";  
a.push(3);  
console.log(a); // Skriver ut "[1, "two", 3]"  
  
var b = [1, "two", 3]; // Likadan som a!  
console.log(b); // Skriver ut "[1, "two", 3]"
```

Datatyper

Några bonustyper

- null
- undefined
- function

Lite mer avancerad Javascript

Funktioner

Funktioner är återanvändbara samlingar av satser, dvs en eller flera rader av kod. De kan ta in *argument* och ge ett *returvärde*.

```
function allan(a, b) {  
  return a + b;  
}
```

Funktioner som variabler

Funktioner kan även vara variabelvärden:

```
var anka = function (a, b) {  
    return a + b;  
};  
console.log(anka(1, 2)); // Skriver ut "3"
```

Det gör att de kan anropas lite hur som helst:

```
var b = {  
    name: "addition"  
};  
b["add"] = anka;  
console.log(b.add(2, 3)); // Skriver ut "5"
```

Variabeltyper – var, let och const

Variabler kan deklareras på tre olika sätt:

- **var** kan omdeklaras och uppdateras. Äldre variant.
- **let** kan uppdateras men inte omdeklaras.
- **const** kan varken uppdateras eller omdeklaras – den är konstant.

Både **let** och **const** är block-bundna, **var** är det inte.

var

```
var a = 1; // Kan omdeklareras  
var a = 2; // Helt okej, men konstigt  
  
if (a == 2) {  
    var language = "sv";  
    console.log("We're using " + language);  
}  
console.log(language); // Kommer att skriva ut "sv"
```

let

```
let a = 1; // Kan inte omdeklareras
let a = 2; // Fungerar inte, du kommer att få ett fel
a = 2;      // Fast det här är helt okej!

if (a == 2) {
  let language = "sv";
  console.log("We're using " + language);
}
console.log(language); // Ger ett fel, ReferenceError
```

const

```
const a = 1; // Kan inte omdeklaras  
a = 2;      // Fungerar inte, redan deklarerad!
```

```
const teacher = {  
  name: "Johan",  
  salary: 10000,  
  colleague: {  
    name: "Anton",  
    younger: true  
  }  
};
```

```
teacher = "Sebastian"; // Går inte, redan definierad!  
teacher.name = "Sebastian"; // Helt okej, dock!
```

Synlighet

Se “whiteboarden”!

Closures

```
/*
 * En closure-funktion kopplar samman data med en (eller flera) funktioner.
 * Det påminner lite om hur en objekt-orienterad klass fungerar.
 */
function create() {
  var name = 'Ujjal Kaur';    // Denna ligger i samma scope som printName()
  function printName() {
    console.log(name);        // Samma scope = åtkomst
  }
  return printName;
}
var f = create();              // Detta kommer att generera en ny closure åt oss
f();
```

Closures, objekt och annat kul

```
// Ett lite mer avancerat exempel som beskriver en vanlig morgon i det
// holmbergska hemmet efter lite för lång jobbnatt.
var snooze = (function () { // Här tilldelar vi snooze en closure direkt
    var snoozes = 0;        // Det här syns inte utåt = privat

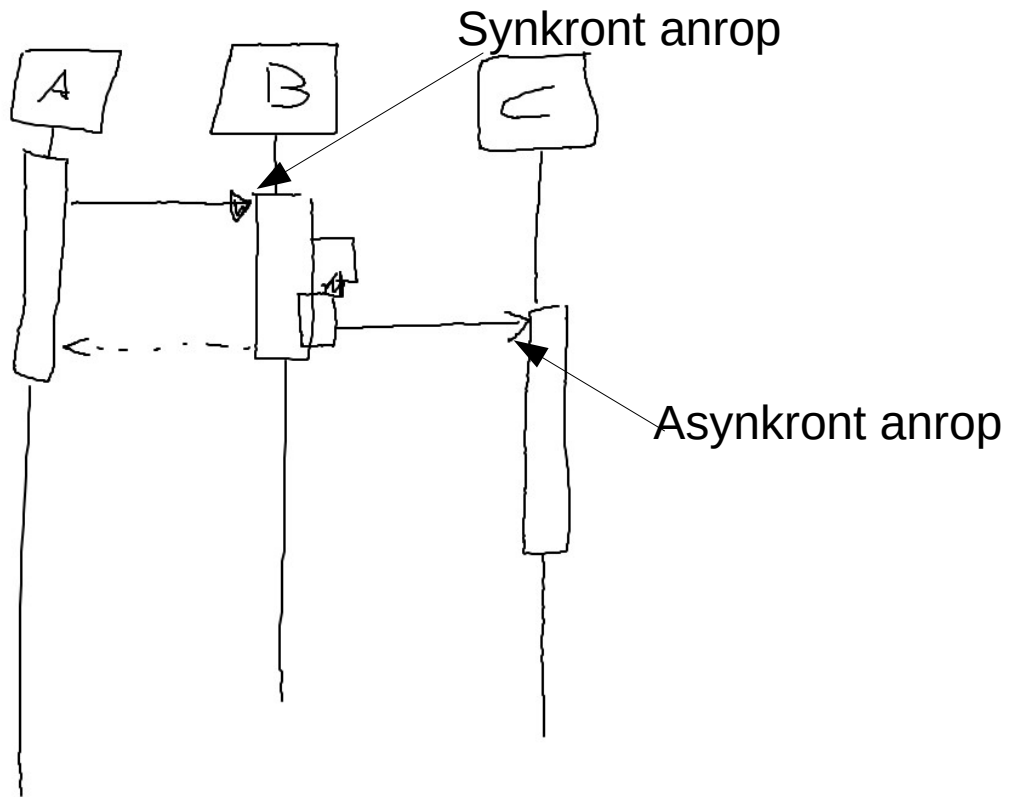
    function snooze(minutes) { // Den här syns inte utåt = privat
        snoozes += minutes;
    }

    return {                // Det är det här objektet med funktioner som syns utåt
        justOneMore: function() {
            snooze(5);
        },
        howLongDidISnooze: function() {
            console.log(snoozes + " minutes");
        }
    }
})();

snooze.justOneMore();
snooze.justOneMore();
snooze.justOneMore();
snooze.howLongDidISnooze();
```

Asynkron programmering

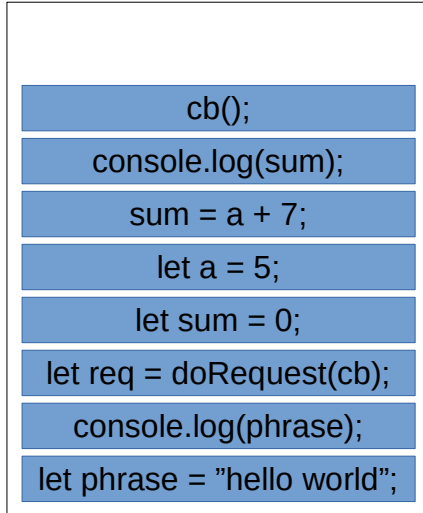
I Javascript händer saker inte alltid i den följd som du förväntar dig om du är van vid Python eller Java.



Asynkron programmering

(Notera att anropsstacken egentligen jobbar med hela funktioner, den här enklare koden är tänkt som en illustration)

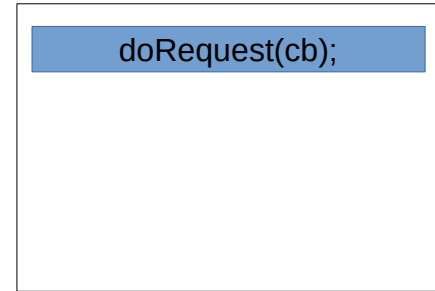
Anropsstack



Javascript-kod

```
let phrase = "hello world";  
console.log(phrase);  
let req = doRequest(cb);  
let sum = 0;  
let a = 5;  
sum = a + 7;  
console.log(sum);
```

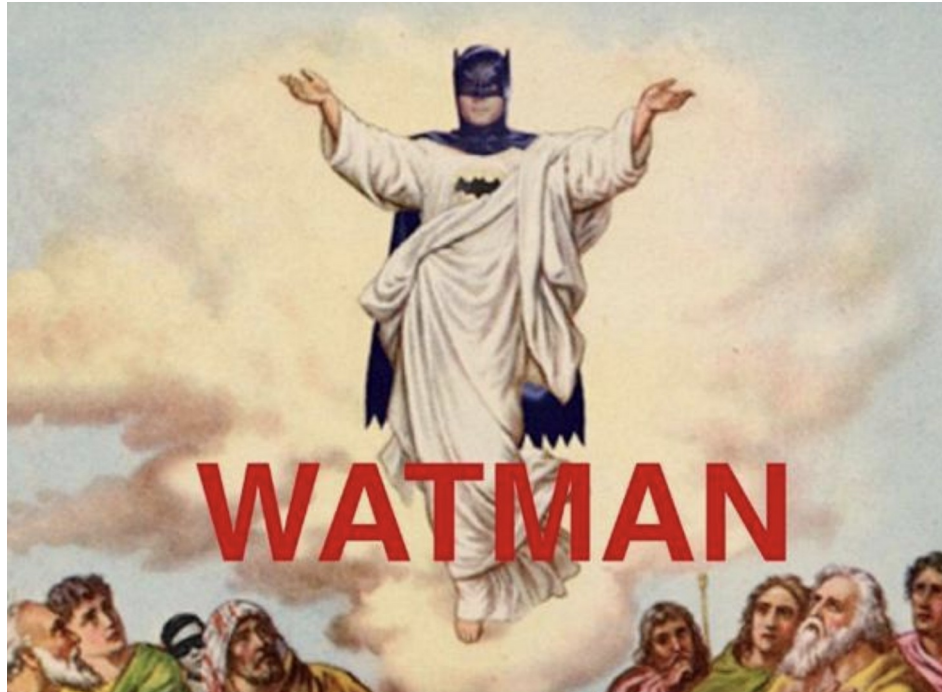
Externa anrop



Händelsekö



Batman!



<https://www.destroyallsoftware.com/talks/wat>