JAVASCRIPT 2

THE RETURN

DAGENS AGENDA

Förmiddag

Kursens innehåll

Repetition av JS1

Eftermiddag

Praktiska JS-uppgifter

Arrayer & Objekt

FOKUS

Fördjupning inom vanilla JavaScript

- this och objekt
- Prototyper
- Designmönster
- AJAX

TILLÅTET

jQuery får användas för att lösa den andra examinerande.

Men inga häftiga ramverk.

Förslag: Introduktion till `nodes` ekosystem och grundläggande om byggverktyg

https://github.com/FEND16/javascript2/pull/1

VAD SÄGER DEN ALLSMÄKTIGA KURSPLANEN

JSDOC

Ni kommer att se kommentarer som ser ut såhär:

```
/**
 * Function that says your name with hello before
 * @param  {String} name Your name
 * @return {String}      Your name with hello in fro
 */
function hello(name){
    return 'Hello' + name;
}
```

ES6 / E2015

Ni kommer att se kod som ser ut såhär:

```
const multiply = (a, b) => {
  return `Summan av talen är: ${a * b}`;
}
```

Jag kommer att använda ES6-syntax

För att det är nice.

ES6

let

const

arrow function

Template Literals

LOOPING

Jag kommer ibland att ersätta `for` -loopen med:

`Array.map()`

`Array.reduce()`

`Array.filter()`

`Array.forEach()`

Inget krav men väldigt bra funktioner att kunna

EXAMINATION

Två examinerande inlämningar

Designmönster: skapa en Movie Database

Fokus på att skriva tydlig och fräsch kod

AJAX: Hämta data från öppna APIer

Fokus på att använda AJAX

MAN SKA ÄVEN VISA ATT MAN FÖRSTÅR

PROTOTYPKEDJAN

Kunna skriva läsbar och strukturerad kod

Kunna använda ES6

REPETITION

BUILT IN TYPES

- Number
- String
- Bool
- Object
- null
- undefined

Ref: YDKJ - Values & Types

FALSY VALUES

- ""
- 0, -0, NaN
- null
- undefined
- false

Ref: YDKJ - Truthy & Falsy

TYPE COERSION

JavaScript är ett dynamiskt typat språk

Det finns bara `var` och alla värden konverteras implicit om man inte säger annat

Viktigt: `===` vs. `==`

Ref: YDKJ - Type Coercion

FOR

```
for(var i = 0; i < 10; i++){
  console.log(i);
}
```

```
for(start; duration; incrementation){
  console.log();
}
```

Ökningen behöver inte alltid vara `i++` men är oftast det.

Ref: YDKJ - Loops

IF/ELSE/ELSE IF

```
if(condition){
  //Run code
}
```

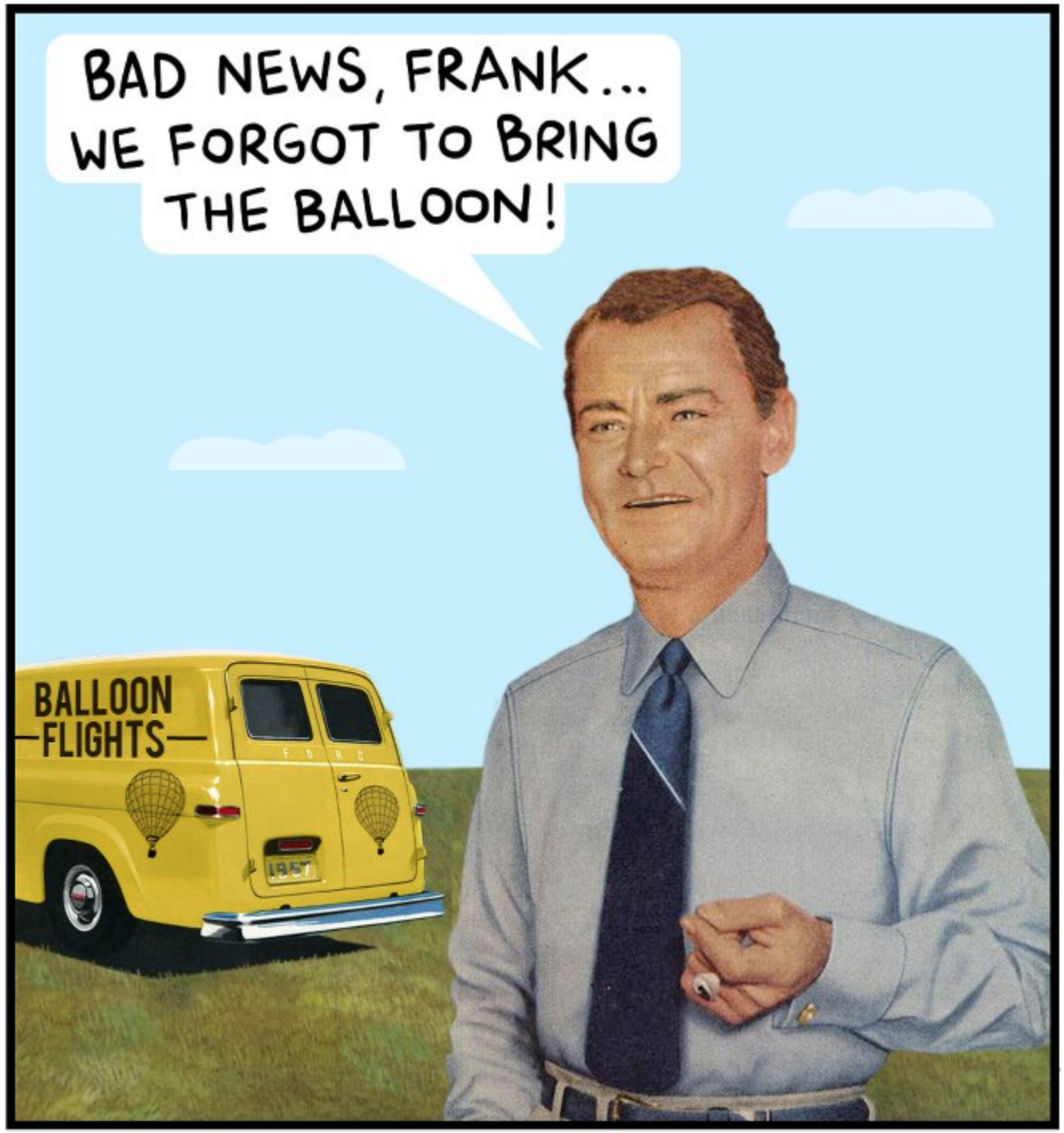
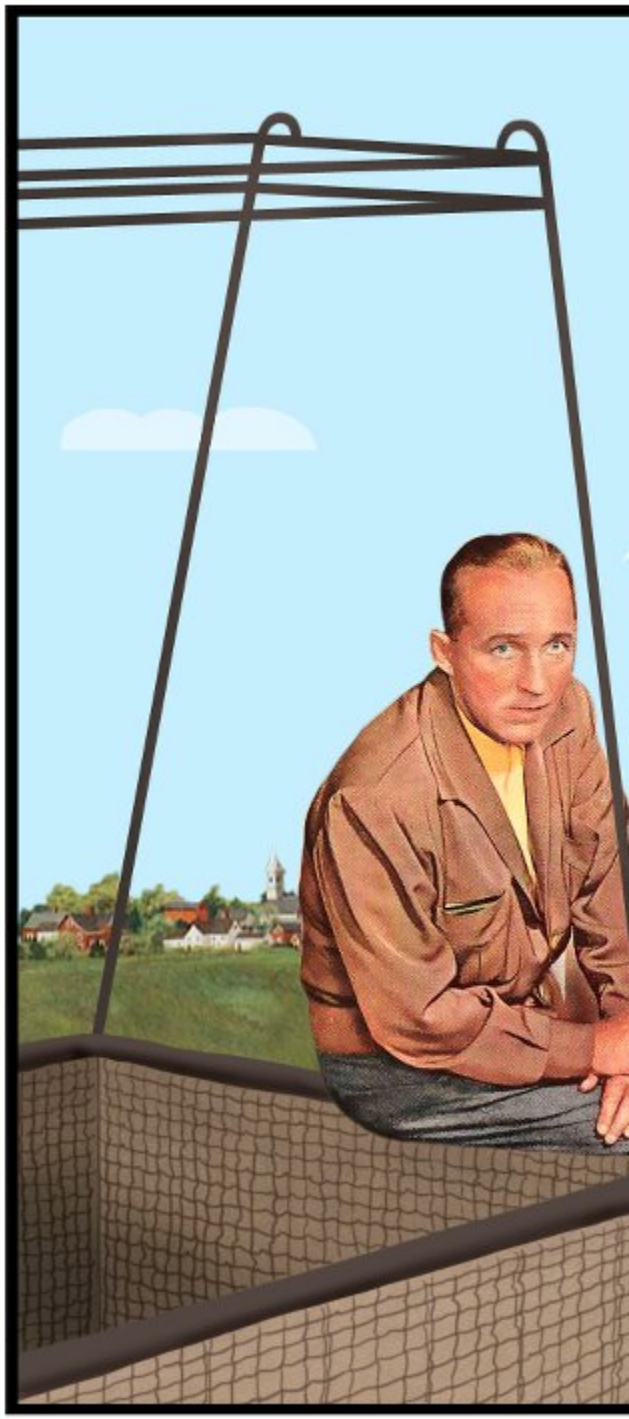
Om ett villkor inte uppfylls kommer koden aldrig att nås.

Därför viktigt att att tänka på i vilken ordning **if-satserna** positioneras.

Ref: YDKJ - Conditionals

NESTED HELL

```
if(condition){
  if(condition){
    if(condition){
      if(condition){
        eturn true;
    }
  }
}
```


BAD NEWS, FRANK... WE FORGOT TO BRING THE BALLOON!
BALLOON FLIGHTS


ABSTRAHERA

Känner man att koden är för rörig på vissa ställen:

abstrahera

Lägg delar av koden i en ny funktion

```
if(condition){
  doMoreCode();
}
```

Många och små funktioner är inget problem, bara man är tydlig med vad funktionen gör.

FUNCTION DECLARATION

```
function sayHello(){
  return "Hello!";
}
```

Funktionen hoistas : skjuts upp i scopet och kommer att vara tillgänglig i hela scopet

Ref: YDKJ - Functions as values

SCOPE

```
//Global scope
function sayHello(){
  //Function scope
  return "Hello!";
}
```

En variabel i en funktion är inte tillgänglig utanför funktionen men en variabel i global scope är tillgänglig överallt. (Dålig grej)

Ref: YDKJ - Function Scopes

FUNCTION EXPRESSION

```
var sayHello = function(){
  return "Hello!";
}
```

Spara en anonym function i en variabel. Fungerar på samma sätt som en "vanlig funktion".

Det spelar oftast ingen roll vilken du använder.

Ref: SO - FD vs. FE

NO RETURN

INGEN funktion MÅSTE returnera någonting

```
function hello(){
  console.log("Hello");
}
```

Funktionen loggar ENBART ut värden. Vi kan inte använda funktionen till något annat.

Funktioner BORDE returnera **ETT** värde/array/objekt

RETURN

```
function hello(){
  return "Hello";
}
```

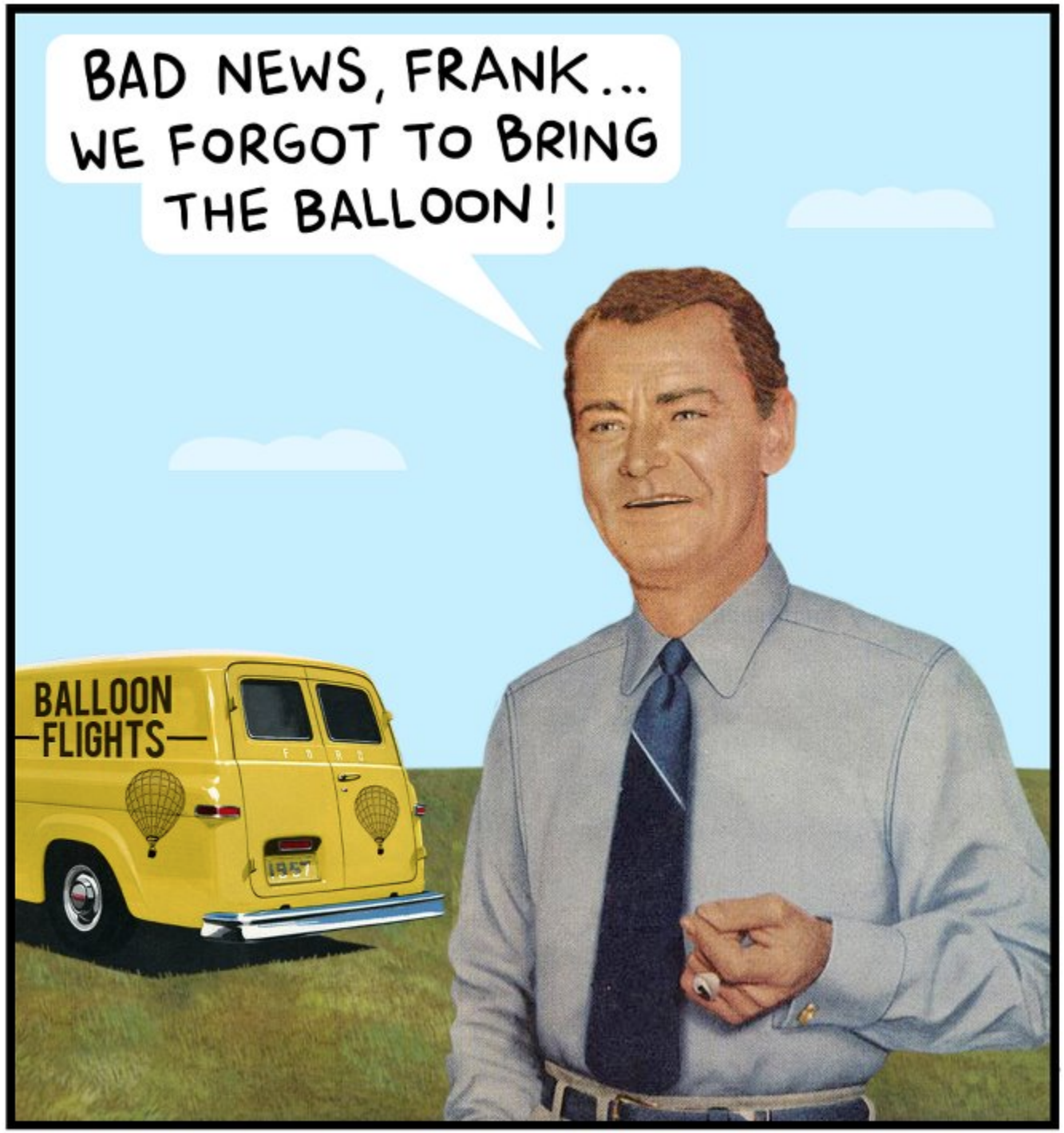
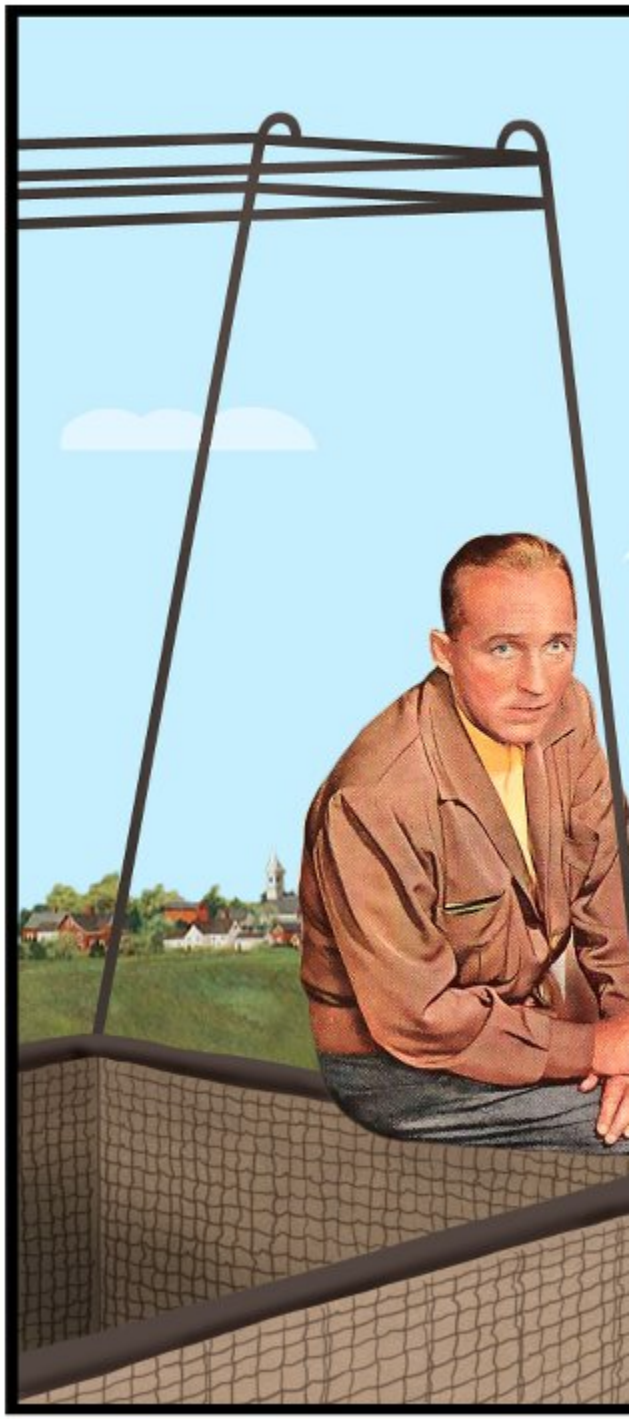
`console.log(hello());`

`hello() === "Hello"`

return betyder att funktionen är klar, inget mer ska hända. Gå tillbaka till stacken.

FUNKTIONERNAS FUNKTION

- Funktioner ska vara korta
- Funktioner ska oftast bara göra EN grej
- Det finns dock ingen funktions-polis
- Ibland kan man inte/har inte tid att göra världens bästa funktioner.


BAD NEWS, FRANK... WE FORGOT TO BRING THE BALLOON!
BALLOON FLIGHTS


ARRAYER

```
var array = [0, 5, 6, 0, 12];
```

Arrayer är samlingar av variabler.

Index, startplatsen är 0

Även om längden är 3 är sista indexet 2: 0 1 2

Ref: YDKJ - Arrays

ARRAYENS INNEHÅLL

Arrayen kan innehålla alla olika värden

```
var array = [ null, true, "string", 0, "0", [
```

Försök att hålla arrayen fylld med samma sorts typ

OBJECT

Objekt är datastrukturer som använder key/value

Vanligaste sättet att skapa objekt är Object literal

```
var object = {
  property: value,
  property2: value
};
```

Ref: YDKJ - Objects

OBJECT

```
var recipe = {
  name: 'Mandelkubb',
  characteristics: 'Sjukt torr, blötlägg inna
  ingredients: []
}
```

Ett objekts property value kan vara ett annat objekt, eller en array.

Allting i JavaScript är objekt så allting kan lagras i ett objekt.

LOOPA OBJEKT

```
for (var property in obj){
  console.log(property, obj[prop]);
}
```

Vi kommer åt ett objekts värden med dot-notation:

obj.property

Eller som här: **obj[prop]**

ARRAY OF OBJECTS

```
var arr = [
  { name: "Bobbo"},
  { name: "Kloppsky"},
  { name: "Charles VII"}
];
```

DOM-MANIPULATION

```
var el = document
  .getElementById('clicky');
```

```
var tagList = document
  .getElementsByTagName('li');
```

```
var classes = document
  .getElementsByClassName('btn-default');
```

Ref: DOM-manipulation

DOCUMENT.CREATEELEMENT()

```
var el = document.createElement('div');
```

argumentet specificerar vilken typ av element som ska skapas.

funktionen **createElement()** tillhör document

APPENDCHILD()/REMOVECHILD()

- `parent.appendChild(childElement)`
- `parent.removeChild(childElement)`

Ref: DOM-manipulation

.INNERHTML

.innerHTML byter ut allt innehåll innanför taggarna på diven

.appendChild lägger till ett nytt child innanför taggarna på diven

Ska du uppdatera en hel lista så kötta **.innerHTML**

EVENTS

```
element.
  addEventListener('click', function(){
    console.log('You clicked!');
});
```

Detta är att föredra

allting samlat i js-filen

```
element.onclick = function(){
  console.log('You clicked!');
}
```

```
<button onClick="clickFunction()">Click<buttc
```

Jag gillar inte den här metoden.

Men gör som ni vill.

Men inte såhär tack.

CLASSLIST

Vanligaste sättet att animera eller dölja/visa element genom att toggla en klass

```
el.addEventListener('click', function(){
  div.classList.toggle('visible');
  }
```

- classList.add('class');
- classList.remove('class');
- classList.toggle('class');
- classlist.contains()

Ref: MDN - classList

ÖVNING: WORLD STATS

Loopa igenom en array med information om olika länder

Varje land är ett eget objekt med olika egenskaper

Sålla och sortera

GitHub: 01_world_stats @ javascript2