DOM-manipulation

Manipulation at its best

ZoCom

DOM

- DOM = Document Object Model
- Innehåller alla HTML-element i en trädbaserad datastruktur
- Ligger i ett objekt som heter **document** (testa och skriv **document** i konsolen)
- Har en hel del egenskaper och metoder som går att använda
- Varje HTML-element är ett eget objekt med egenskaper och metoder

DOM-selektorer


```
let elem = document.getElementById('myId');
```

Hämta ett specifikt element med detta ID

DOM-selektorer

```
let elem = document.getElementsByClassName('myClass');
```

Returnerar en lista med alla element med detta klassnamn

DOM-selektorer

```
let elem = document.getElementsByTagName('p');
```

Returnerar en lista med alla element med denna HTML-tag

DOM-selektorer

```
let elem = document.querySelector(cssSelector);
```

Returnerar det första element den hittar med den selektorn

DOM-selektorer

```
let elem = document.querySelectorAll(cssSelector);
```

Returnerar en lista med alla element den hittar

med den selektorn

Metoder

document

Metoder

- **document.createElement(*HTML-tagg*)** - skapar ett nytt HTML-element
- **appendChild()**, **removeChild()** - lägger till respektive tar bort ett HTML-element
- Observera att **createElement()** inte lägger till element i DOM:en
- Använd detta när du behöver dynamiskt kunna skapa upp HTML-element

Show me some code!

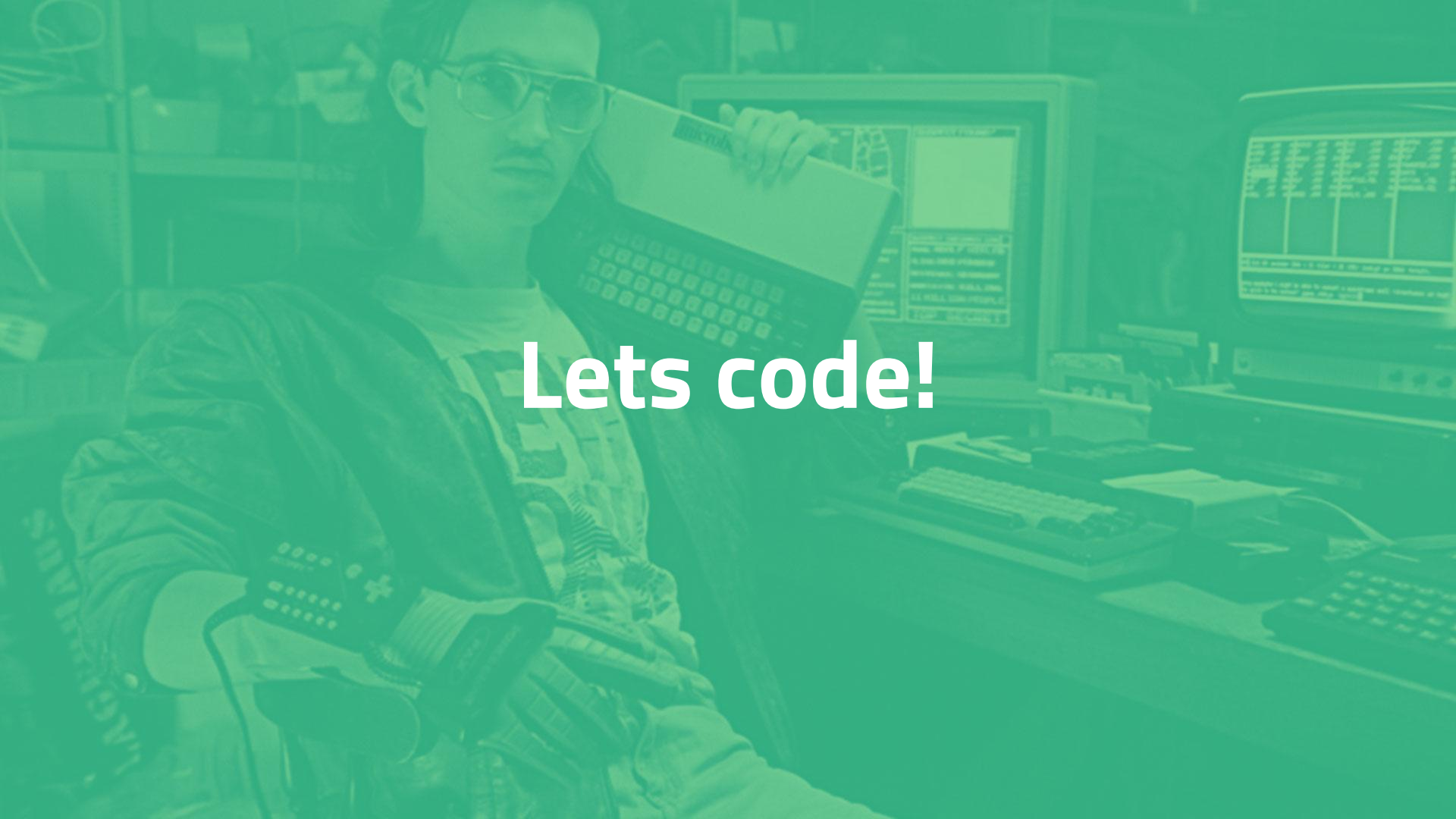
```
let bodyElem = document.querySelector('body');

let heading = document.createElement('h1');
heading.innerHTML = 'Hej';

bodyElem.appendChild(heading);
```

Metoder

På element

Metoder

- **.innerHTML** - för att komma åt innehållet i en HTML-tagg
- **.className/.classList** - för att hämta samt lägga till/ta bort css-klasser
- **.setAttribute(namn, value)/.getAttribute(namn)** - för att lägga till eller hämta ett attribut som tex. *id*, *class*, *type=”text”* etc.
- **.style** - för att modifiera CSS properties


Lets code!

Events

- JavaScript är vad man säger ett event-drivet språk då mycket ker på användarens input
- För att veta när användaren har klickat med musen eller tryckt på exempelvis enter-tangent kan vi använda oss av **events**
- Vi sätter en **eventListener** på ett HTML-element för att prenumerera på uppdateringar och definierar en funktion som ska köras

Events

```
let buttonElem = document.querySelector('button');
buttonElem.addEventListener('click', function(event) {
    //Körs när ett klick-event körs
});
```

Events

```
let buttonElem = document.querySelectorAll('button');
buttonElem.addEventListener('click', function(event) {
    //Körs när ett klick-event körs
});
```

Ser ni några eventuella problem med detta?

Lets diskutera

Diskutera två och två, vad händer här?

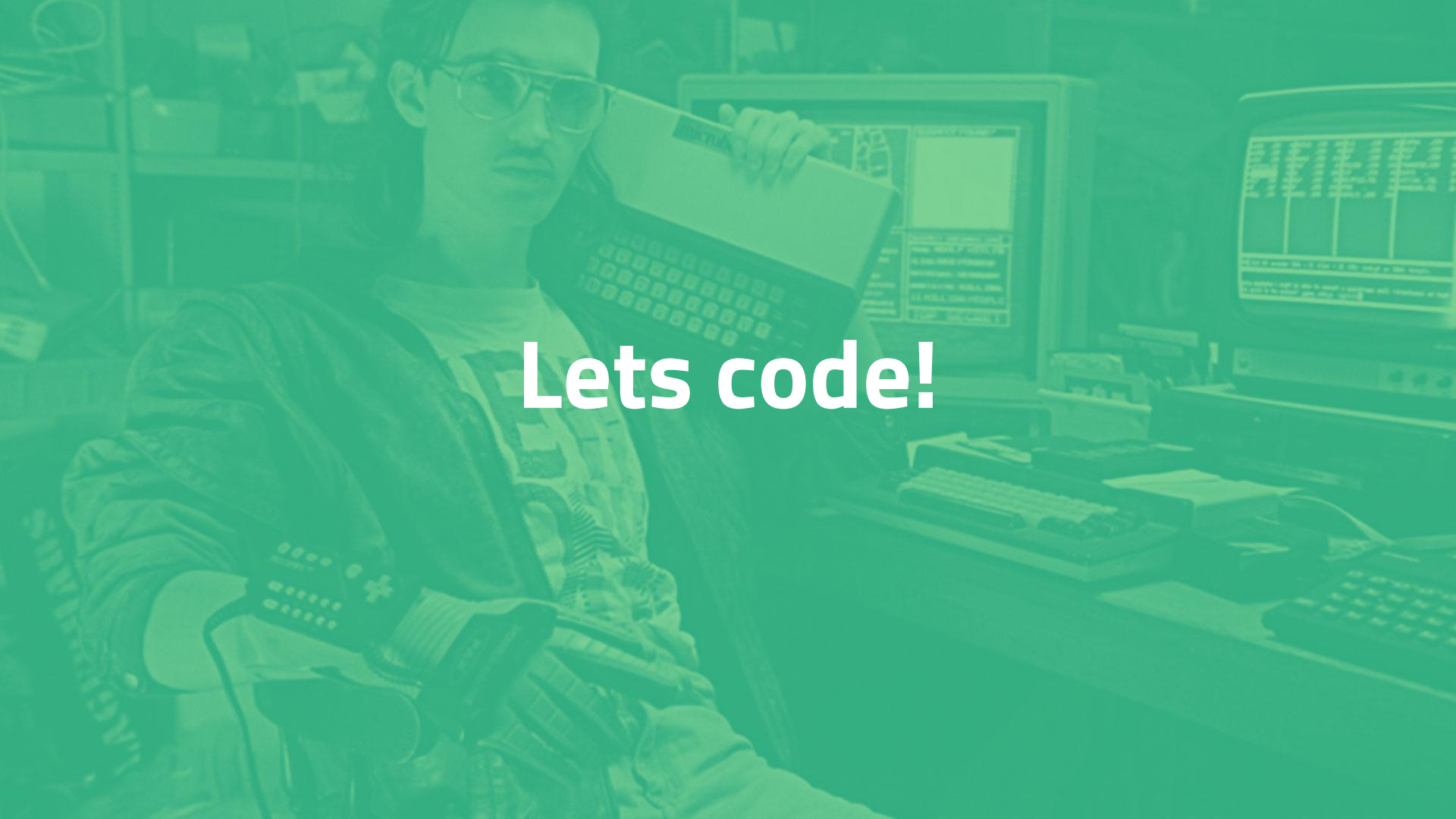
Vad händer?

```
let listElem = document.querySelector('ul');

for(let i=0;i<2;i++) {
    let listItem = document.createElement('li');
    listItem.innerHTML = i;

    listElem.appendChild(listItem);
}
```

Unobtrusive JavaScript

En generell approach till JavaScript på webben

- **Separation of functionality** - separera HTML, CSS och JS
  - Undvik JavaScript i HTML
  - Exempel: Använd inte **onclick** eller liknande utan håll er till **eventListeners**.
- **Graceful degradation** - om något i JavaScript kraschar så kan sidan fortfarande visas
- Läs mer här och här


Lets code!