

Systemutveckling

PHP

Föreläsning 16 - JavaScript

Mikael Olsson
mikael.olsson@emmio.se
076-174 90 43



Dagens ämnen

- Datatyper
- Jämförelseoperatorer
- Funktioner
- DOM
- Array-funktioner
- jQuery
- JSON
- API
- Node.JS
- npm
- webpack
- Continuous Integration

Vad är JavaScript?

- Scriptspråk som används för att skapa dynamiska hemsidor
- Exekveras i webbläsarens JavaScriptmotor, behöver inte kompileras
- Standarden av JavaScript kallas ECMAScript.

Variabler

- En del språk kräver att man berättar vilken typ av data variabeln innehåller, t ex sträng, tal osv. De språken kallas typade eller starkt typade.
- JavaScript är ett svagtypat språk, vi behöver inte berätta vilken typ av information vi vill spara i en variabel.

Typning

- Statiskt typade språk ex:

```
int x = 5;  
x = "hej"; //kompileringsfel!
```

- Dynamiskt (checkas i run-time) & löst (går att byta) typade språk ex:

```
var x = 5; //x är av typen int  
x = "hej"; //x är nu av typen string, typen  
ändras i run-time
```

Datatyper

- `number` för nummer, både heltal och decimaltal.
- `string` för strängar/texter/bokstäver.
- `boolean` för sant/falskt.
- `null` för okända värden.
- `undefined` för värden som inte har tilldelade värden.
- `object` för mer komplexa datastrukturer.
- `symbol` för unika identifierare.
- Array räknas som objekt i JavaScript.
- <https://javascript.info/types>

Variabler - deklaration

```
let firstName = "Mikael"; //firstName är en sträng "Mikael"
```

```
let lastName = ""; //lastName är en tom sträng
```

```
let age; // age är undefined
```

```
lastName = "Olsson"; //lastName är "Olsson"
```

Tilldelning

- `var` - deklarerar en variabel, ev med ett värde. Variabeln finns globalt eller inom en hel funktion.
- `let` - deklarerar en variabel som alltid har ett lokalt scope, ev med ett värde. Variabeln existerar inte utanför blocket, statementet eller uttrycket som det skapades i.
- <http://jsfiddle.net/9ghwc5st/>
- `const` - deklarerar en konstant som inte kan ändras

Jämförelseoperatorer

- "truthy" vs "falsy" jämförelser:
- ==
- !=
- <
- <=
- >=
- >
- vanlig jämförelse mellan datatyper
- ===
- !==
- The identity operator returns true if the operands are strictly equal with no type conversion.
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Comparison_Operators

Aritmetiska operatorer

- Användbara aritmetiska operatorer

+

-

*

/

%

++

--

+=

Aritmetiska operationer - tal

```
let a = 1;
```

```
let b = 2;
```

```
let c = a + b; // c = 3
```

Aritmetiska operationer - sträng

+ används för att konkatenera (slå ihop) strängar

```
var firstName = "Mikael";
```

```
var lastName = "Olsson";
```

```
var name = firstName + lastName; // name = "MikaelOlsson"
```

```
var fullName = firstName + " " + lastName; // fullName = "Mikael Olsson"
```

console.log() och felhantering

console.log(), används för att skriva ut variabler eller element i konsolen i webbläsaren

Användbart under utvecklingen och vid felhantering

```
var firstName = "Mikael";  
  
console.log(firstName); //Skriver ut "Mikael"  
console.log("firstName = " + firstName); //Skriver ut  
"firstName = Mikael"
```

Interaktion med HTML/CSS

- Importeras ungefär som CSS-filer i HTML-koden
- Kan importeras i head eller nära `</body>`, för bättre performance.
- Kan "nå" HTML-elementen och ändra dem.
- JavaScript kan "triggas" av användaren genom HTML-sidan

Interaktion med HTML/CSS

Importerera JavaScript-filer i HTML

```
<!DOCTYPE html>
<html>
  <head>
    <script src="script1.js"></script>
  </head>
  <body>

    <script src="script2.js"></script>
  </body>
</html>
```

Interaktion med HTML/CSS

Nå HTML element från JavaScript-filen

```
document.addEventListener("DOMContentLoaded", function(event) {  
    var headline = document.getElementById("headline");  
    var text = headline.innerHTML;  
});
```


Interaktion med HTML/CSS

Ändra HTML element från JavaScript-filen

```
document.addEventListener("DOMContentLoaded", function(event)
{
    var headline = document.getElementById("headline");
    var oldText = headline.innerHTML;
    headline.innerHTML = "New Headline";
    headline.style.color = "#0000ff";
});
```

Interaktion med HTML/CSS

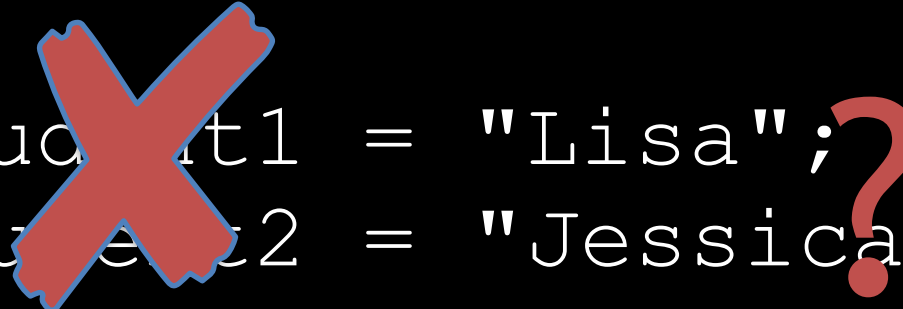
Hämta information från inputfält

```
document.addEventListener("DOMContentLoaded", function(event) {  
    var name = document.getElementById("name").value;  
    var acceptRules = document.getElementById("rules").checked;  
});
```

Värden som hör ihop

- Hur ska man göra om man har flera studenter?
- Begränsande!

```
var student = "Kalle";  
var student1 = "Lisa";  
var student2 = "Jessica";  
var student3 = "Lisa";
```

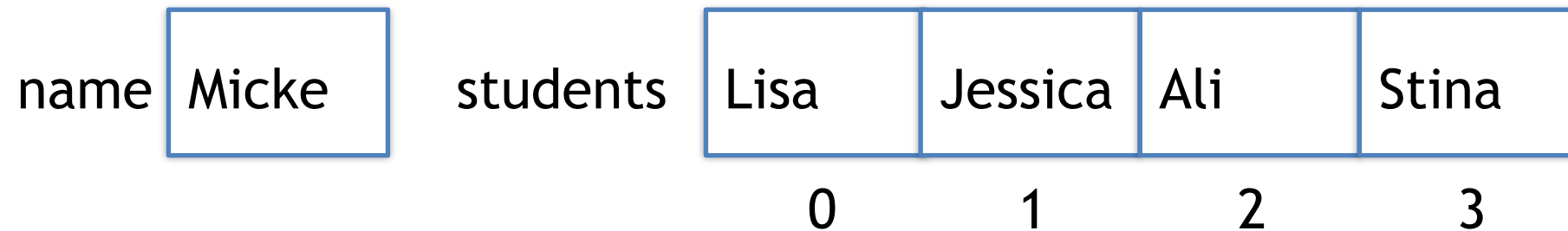


Array

En array är en speciell datatyp som kan innehålla flera värden.

```
var students = ["Lisa", "Jessica"];  
  
// Kan stå på flera rader  
  
var students = [  
    "Lisa",  
    "Jessica"  
];
```

Array



- En array är en speciell datatyp som kan innehålla flera värden.

```
var name = "Micke";  
var students = [  
    "Lisa",  
    "Jessica",  
    "Ali",  
    "Stina"  
];
```

Array

- *Uppgift:* Skapa en array och skriv ut varje värde i den.
- <http://jsfiddle.net/>

```
var studenter = [  
  "Lisa",  
  "Jessica",  
  "Ali",  
  "Stina"  
];  
  
console.log(studenter[0]);  
// osv...
```

Lisa



Avancerade variabler

- Saker i riktiga världen består ofta av olika variabler.
- Hänger de ihop på något sätt?

```
var name = "Micke";  
var age = 42;  
var shoe_size = 43;
```

Objekt

- *Objekt* är variabler med "undervariabler".

```
▶ {name: "Micke", age: 42, shoe_size: 43}
```

```
>
```

```
var person = {};  
  
person.name = "Micke";  
person.age = 42;  
person.shoe_size = 43;  
  
console.log(person);
```

```
var person = {  
  name: "Micke",  
  age: 42,  
  shoe_size: 43  
};  
  
console.log(person);
```


Avancerade variabler

- *Uppgift:* Skapa ett objekt som håller reda på en *kurs* med några egenskaper.

```
var person = {  
  name: "Micke",  
  age: 42,  
  shoe_size: 43  
};
```

Funktions beståndsdelar

- Funktioner är en kod-block som utförs när vi vill att den ska utföras.
- Funktioner kan ta emot data i så kallade parametrar.
- Funktioner kan kallas av JavaScript-kod eller när användaren gör något, t.ex. klickar på en knapp.

Funktioner

- Funktionens beståndsdelar

```
function addNumbers(a, b) {  
  console.log(a + b);  
}
```

```
addNumbers(2, 2);
```

- Namn
- Parametrar
- Innehåll
- Anrop

4

>

Funktioner

- En funktion kan även returnera ett värde.

```
function addNumbers(a, b) {  
  return(a + b);  
}  
  
var result = addNumbers(2, 2);  
  
console.log(result);  
console.log(addNumbers(3, 3));
```

4
6
>

Funktioner

- Ser vi något mönster?
- Vad händer om vi måste ändra i formeln?

```
var temp_f = [80, 75, 88];  
var temp_c;  
var tmp;
```

```
tmp = temp_f[0];  
temp_c = (5/9) * (tmp-32)  
console.log(temp_c);
```

```
tmp = temp_f[1];  
temp_c = (5/9) * (tmp-32)  
console.log(temp_c);
```

```
tmp = temp_f[2];  
temp_c = (5/9) * (tmp-32)  
console.log(temp_c);
```

Funktioner

```
function toCelcius(fahrenheit) {  
    var temp_c = (5/9) * (fahrenheit-32);  
    console.log(temp_c);  
}
```

```
var temp_f = [80, 75, 88];  
var tmp;
```

```
tmp = temp_f[0];  
toCelcius(tmp);
```

```
toCelcius(temp_f[1]);  
toCelcius(temp_f[2]);
```

26.666666666666668

23.88888888888889

31.111111111111114

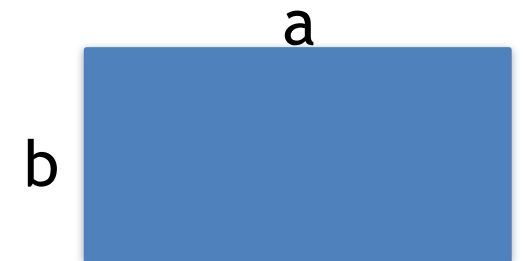
>

Funktioner

- *Uppgift:* Gör en funktion som räknar ut arean av en rektangel och returnerar resultatet.
- <http://jsfiddle.net/>

```
function toCelcius(fahrenheit) {  
  var temp_c = (5/9) * (fahrenheit-32);  
  console.log(temp_c);  
}
```

```
toCelcius(80);
```



Area = a * b

Tilldelning

- `var` - deklarerar en variabel, ev med ett värde. Variabeln finns globalt eller inom en hel funktion.
- `let` - deklarerar en variabel som alltid har ett lokalt scope, ev med ett värde. Variabeln existerar inte utanför blocket, statementet eller uttrycket som det skapades i.
- <http://jsfiddle.net/9ghwc5st/>
- `const` - deklarerar en konstant som inte kan ändras

Mer om array

```
var students = [  
    "Lisa",  
    "Jessica",  
    "Ali",  
    "Stina"  
];  
  
let l = students.length;  
for (let i = 0; i < l; i++) {  
    console.log(students[i]);  
}
```

Mer om array

```
var students = [  
    "Lisa",  
    "Jessica",  
    "Ali",  
    "Stina"  
];  
  
students.forEach(function(element) {  
    console.log(element);  
});
```

Lägga till värde i array

```
var students = [  
    "Lisa",  
    "Jessica",  
    "Ali"  
];  
  
students.push("Stina");
```

Ta bort värde från array

- pop - tar bort från slutet av en array och returnerar värdet.
- shift - tar bort från början av en array och returnerar värdet.
- splice - tar bort från en array för ett specifikt index och returnerar värdet / värdena.
- filter - returnerar en ny array med filtrerade element från en array.
- <https://love2dev.com/blog/javascript-remove-from-array/>

Objekt

Man kan ha objekt i arrayer.

<http://jsfiddle.net/2r1ph08o/>

```
let students = [];  
  
let person = {  
  name: "Micke",  
  age: 42,  
  shoe_size: 43  
};  
students.push(person);
```

```
person = {  
  name: "Jessica",  
  age: 25,  
  shoe_size: 36  
};  
students.push(person);  
  
console.log(students);
```

Grupppuppgift

- I grupper om 3-4 personer ska ni bygga en site där användare ska kunna registrera sig och spela yatzy.
 - Det ska finnas ett Yatzy-formulär för upp till fyra spelare.
- Extrauppgifter:
 - Användare ska kunna logga in/registrera.
 - Användaren ska ha en profil-sida
 - Det ska finnas en lista med senaste matcherna (hitta på resultat t v).
 - Lägg upp ett gemensamt repo på GitHub.

Yatzy

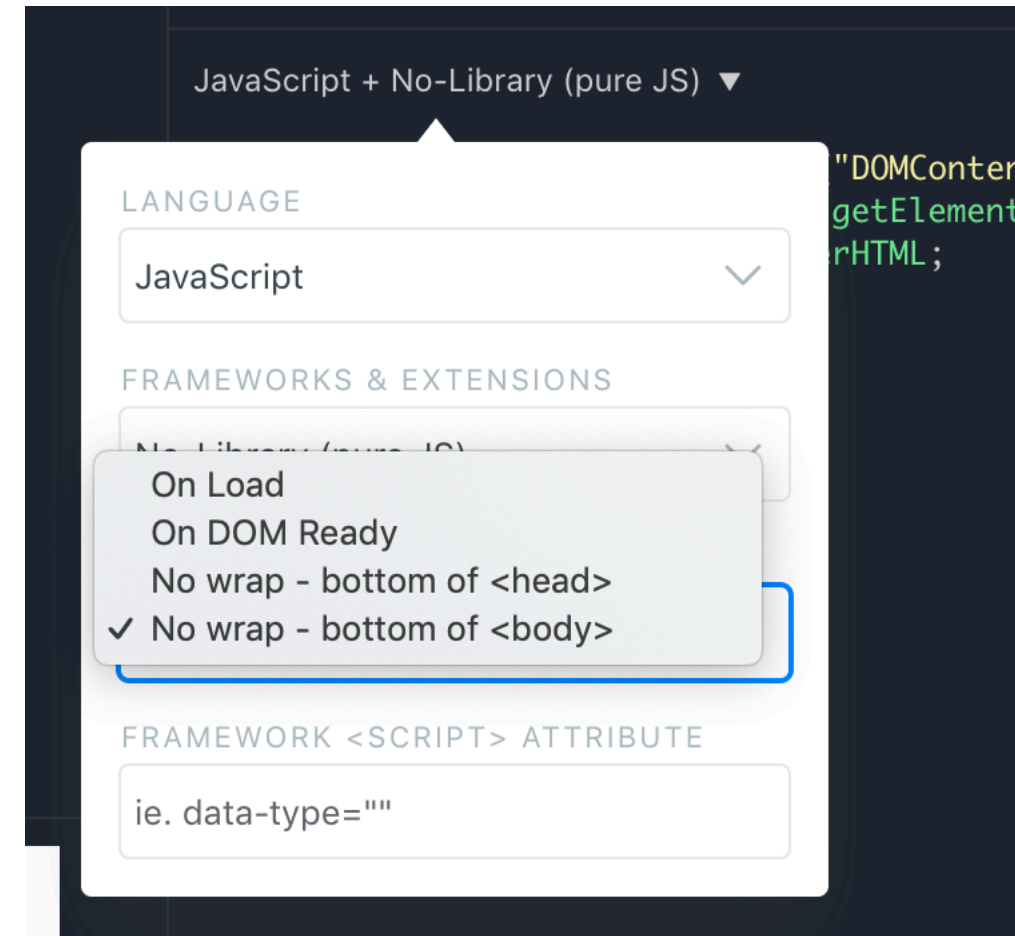
- Diskutera i era grupper hur en lämplig datastruktur för spelare och resultat skulle kunna se ut.
- Återvänd och diskutera i klassen.
- Implementera.

DOM

- Document Object Model - detta är JavaScriptets objektrepresentation av html-koden.
- Plattforms- och språkoberoende gränssnitt som ger programspråk möjligheten att dynamiskt läsa och uppdatera ett dokument innehåll, struktur och formatering.

jsfiddle.net

- JsFiddle ger möjlighet att bestämma när scriptet ska köras.
- När dokumentet har laddats.
- När DOM:en är klar.
- Lägg skriptet i `<head>`.
- Lägg skriptet precis innan `</body>`.



Interaktion med HTML/CSS

Om du har valt att inte wrappa skriptet kan du själv behöva se till att elementen är skapade.

```
document.addEventListener("DOMContentLoaded", function(event) {  
    // Do stuff  
});
```

Event

- Vad är det för typ av event?

```
<button id="calc">Beräkna</button>
```

```
// - - -
```

```
// Spara beräkna-knappen i en variabel.  
var calc_button = document.getElementById("calc");
```

```
// Ha koll på om någon klickar på beräkna-knappen.  
calc_button.addEventListener("click", function(event) {  
    // Do something.  
});
```

```
// - - -
```

Testa

- Lägg till en knapp som visar en meddelanderuta när ni klickar på knappen.
- Använd den inbyggda funktionen `alert("Meddelande");`
- *Extra uppgift:* Lägg in en input av typ colorpicker. När användaren väljer en färg ska ni uppdatera bakgrundsfärgen på ert dokument (eller något annat) med den färgen med hjälp av JS.

Yatzy

- Ge era inputs id:n, typ "player1_ones".
- När man klickar på Beräkna-knappen ska summan av alla ettor, tvåor, treor, fyror, femmor och sexor beräknas och summan ska uppdateras. Det räcker att göra det för en spelare.
- Skriv ut värde på olika sätt, t ex `innerHTML` om ni har ett element typ `p`, `span`, `td` eller liknande. `value` om ni har en `input`.
- Om spelarens summa är minst 63 poäng ska användaren få 50 poäng i bonus.
- *Extrauppgift:* Gör en funktion som tar en array med fem tal som parameter och returnerar sant om talen innehåller en kåk. (3 av samma + 2 av samma)

Funktioner

- Vilken typ är parametrarna?

Sträng

Funktion

```
document.addEventListener ("DOMContentLoaded", function(event) {  
    var headline = document.getElementById("headline");  
  
    headline.innerHTML = "Lorem ipsum";  
});
```

Funktioner

- Det måste betyda att vi kan ange en "extern" funktion som parameter!

```
function something(event) {  
    var headline = document.getElementById("headline");  
  
    headline.innerHTML = "Lorem ipsum";  
}  
  
document.addEventListener ("DOMContentLoaded", something);
```

Inbyggda funktioner

- JS har en massa inbyggda funktioner, t ex ett matte-bibliotek med matte-funktioner.
- Här är ett exempel på hur man kan slumpa fram ett tal.
- Returnerar ett tal större än eller lika med 0, mindre än 1.

```
var slump = Math.random();  
console.log(slump);
```

0.7126295249855692

0.5267871618023883

0.3054544574128275



Math.random

- Räkna fram ett heltal i ett visst intervall.

```
function getRandomInt(min, max) {  
  // The maximum is exclusive and the minimum is inclusive  
  return Math.floor(Math.random() * (max - min)) + min;  
}
```

Funktioner

- *Uppgift:* Gör ett formulär med ett text-fält och tre knappar.
- När man klickar på någon av knapparna ska en funktion anropas som slumpar fram ett tal och uppdaterar värdet i textrutan med det.
- Det ska vara samma funktion som anropas vilken knapp man än trycker på.



Button 1 Button 2 Button 3

```
var slump = Math.random();  
console.log(slump);
```

Andra event-typer

- `keypress`
- `keydown`
- `keyup`
- `click`
- `mousedown`
- `mouseup`
- `change`
- `submit`
- <https://developer.mozilla.org/en-US/docs/Web/Events>

Default

- Ibland vill man inte att det som normalt händer när ett event utlöses ska hända.

```
calc_button.addEventListener("click", function(event) {  
    event.preventDefault();  
    // Do something.  
});
```

Yatzy

- Skapa en funktion som gör beräkningen för att uppdatera summan.
- Ta bort beräkna-knappen och gör så att summan uppdateras när något av fälten (ettor, tvåor t o m sexor) ändras. Gör det bara för en spelare, ni kommer troligen att vilja lösa detta mer generellt senare.
- Gör ett träningsformulär med fem text-rutor, fem kryssrutor och en knapp. När man klickar ska de rutorna som inte är kryssade få ett nytt tal.

```
var slump = Math.random();  
console.log(slump);
```

getElementById

- Returnerar elementet som matchar.

```
let element = document.getElementById(id);
```

getElementsByClassName

- Returnerar alla element som matchar.

```
// Get all elements that have a class of 'test'
let elements = document.getElementsByClassName('test')

// Get all elements that have both the 'red' and 'test'
// classes
let elements = document.getElementsByClassName('red test')

// Get all elements that have a class of 'test', inside of an
// element that has the ID of 'main'
let elements =
document.getElementById('main').getElementsByClassName('test')
```

getElementsByTagName

- Returnerar alla element som matchar.

```
// Get all elements by the tag  
let elements = document.getElementsByTagName('p');
```


querySelector

- Returnerar det första elementet som matchar.
- `var el = document.querySelector(".myclass");`
- `querySelectorAll` returnerar alla matchande element.

```
let highlightedItems =  
document.querySelectorAll(".highlighted");  
  
highlightedItems.forEach(function(userItem) {  
    deleteUser(userItem);  
});
```

getAttribute / setAttribute

```
// getAttribute
let div1 = document.getElementById("div1");
let align = div1.getAttribute("align");

alert(align); // shows the value of align for the element
               // with id="div1"

// setAttribute
let b = document.querySelector("button");

b.setAttribute("name", "helloButton");
b.setAttribute("disabled", "");
```

innerHTML / textContent

- textContent har bättre prestanda eftersom dess innehåll inte parses som HTML. Dessutom kan textContent förhindra XSS attacker.
- Cross-site scripting (XSS) är ett säkerhetshål som låter någon injicera skadlig kod. Denna kod körs av klienten.
- <https://www.youtube.com/watch?v=T1QEs3mdJoc>

```
// innerHTML
let div1 = document.getElementById("div1");
div1.innerHTML = "New content";
```

parent / children

- If the node has no element children, then children is an empty list with a length of 0.

```
if (node.parentElement) {  
    node.parentElement.style.color = "red";  
}
```

Lambda-funktioner

- Uttryck som skapar funktioner.

- Pre ES6:

```
var anon = function (a, b) { return a + b };
```

- ES6:

```
// Ovanstående exempel:
```

```
var anon = (a, b) => a + b;
```

```
// Eller
```

```
var anon = (a, b) => { return a + b };
```

```
// Om vi bara har en parameter kan vi skippa
```

```
// parenteserna
```

```
var anon = a => a;
```

Lambda-funktioner

// Jämför

```
[1,2,3,4].filter(function (value) {  
    return value % 2 === 0  
});
```

// med:

```
[1,2,3,4].filter(value => value % 2 === 0);
```

<https://www.vinta.com.br/blog/2015/javascript-lambda-and-arrow-functions/>

Map



```
const myArray = [1,2,3,4];

const myArrayTimesTwo = myArray.map((value, index, array) => {
  return value * 2;
});

console.log(myArray); // [1,2,3,4];
console.log(myArrayTimesTwo); // [2,4,6,8];
```

- `map()` -metoden skapar en ny array med resultatet av att anropa en funktion för varje array-element.
- `map()` -metoden anropar den angivna funktionen en gång per element i arrayen i ordning.

Filter



```
const myArray = [1,2,3,4];

const myEvenArray = myArray.filter((value, index, array) => {
  return value % 2 === 0;
});

console.log(myArray); // [1,2,3,4];
console.log(myEvenArray); // [2, 4];
```

- `filter` tar emot samma argument som `map` och fungerar liknande. Enda skillnaden är att callback-funktionen måste returnera `true` eller `false`. Om den returnerar `true` kommer arrayen att behålla elementet, om den returnerar `false` kommer det att filtreras bort.

Reduce



```
const myArray = [1,2,3,4];

const sum = myArray.reduce((acc, currValue, currIndex, array) => {
  return acc + currValue;
}, 0);

const avg = sum / myArray.length;

console.log(avg); // 2.5
```

- reduce tar en array och reducerar den till ett enda värde. Du kan t ex använda det för att räkna ut medelvärdet av alla värden i arrayen.

Map, Filter, Reduce

- <https://medium.com/@joomiguelcunha/learn-map-filter-and-reduce-in-javascript-ea59009593c4>

Styra utseende med JS

- Man kan styra enskilda style properties.
 - `document.getElementById("something").style.backgroundColor = "#ccdd33";`
- Man kan även lägga till eller ta bort klasser.
 - `document.getElementById("div1").classList.add("classToBeAdded");`
 - `document.getElementById("div1").classList.remove("classToBeRemoved");`

jQuery

- <https://code.jquery.com/>
- Inkluderas som ett vanligt js.
 - `<script src="https://code.jquery.com/jquery-3.3.1.min.js`
- Er egen js-fil ska inkluderas efter jQuery för att ni ska kunna använda det. Ändra inget i jQuery-filen.

jQuery

```
$(selector).action()
```

```
$(this).hide()
```

// gömmer aktuellt element

```
$("p").hide()
```

// gömmer alla p-element

```
$(".test").hide()
```

// gömmer alla element med klassen test.

```
$("#test").hide()
```

// gömmer elementet med id test.

jQuery

```
$ (document) .ready (function () {  
    $ ("button") .click (function () {  
        $ ("p") .toggle ();  
    } ) ;  
} ) ;
```

http://jsfiddle.net/emmio_micke/8bawsjv3/1/

http://jsfiddle.net/emmio_micke/8bv0p2fL/1/

http://jsfiddle.net/emmio_micke/3tc7kd6v/2/

http://jsfiddle.net/emmio_micke/7yo4g5Ls/6/

http://jsfiddle.net/emmio_micke/u7rgxL2o/1/

jQuery

```
// A $( document ).ready() block.  
$( document ).ready(function() {  
    console.log( "ready!" );  
});
```

```
// Shorthand for $( document ).ready()  
$(function() {  
    console.log( "ready!" );  
});
```

Fråga om att ta bort

- Det här funkar att göra i era projekt eller i ett fristående projekt.
- Lägg till en länk för varje produkt som låter användaren ta bort produkten. Fråga om användaren är säker först med `confirm`. Lägg därefter på en klass på meddelande som gör bakgrunden röd och gör en fadeout på elementet.
- https://jsfiddle.net/emmio_micke/2vnfdL4b/4/

Kodkvalitet

- Validering och hjälp
 - <https://validator.w3.org/>
 - <http://www.css-validator.org/>
- JSLint är ett analysverktyg som används för att kontrollera om koden klarar kodreglerna för JS.
 - <https://www.jshint.com/>
 - <https://jshint.com/>

JSON

- JSON (JavaScript Object Notation), är ett kompakt, textbaserat format som används för att utbyta data.

- Påminner detta format om något?

```
{  
  "firstName": "Jason",  
  "lastName": "Smith",  
  "age": 25,  
  "address": {  
    "streetAddress": "21 2nd Street",  
    "city": "New York",  
    "state": "NY",  
    "postalCode": "10021"  
  },  
  "phoneNumber": [  
    { "type": "home", "number": "212 555-1234" },  
    { "type": "fax", "number": "646 555-4567" }  
  ],  
  "newSubscription": false,  
  "companyName": null  
}
```

- `var obj = JSON.parse(text);`

API

- När vi gör ett anrop till en webbsida, vad innehåller svaret?
- Kan man få andra resultat?
 - <https://randomuser.me/api/?results=5>
- Application Programming Interface
- Låter oss hämta det data vi behöver i ett format som vi kan bearbeta, vanligen JSON eller XML.
- Ibland kan man behöva autentisera sig i anropet för att få tillgång till data.

Ajax med jQuery

- Asynkron datahämtning, utför instruktioner i bakgrunden, t ex spara information, hämta information.
- Vi kan t ex hämta data i bakgrunden och uppdatera bara delar av webbsidan.
- http://jsfiddle.net/emmio_micke/u8esvgdm/

Uppgift

- <https://randomuser.me/api/>
- Skapa en sida som presenterar en person.
- Vid knapp-klick ska ni hämta data från api:et med ett ajax-anrop och uppdatera data i sidan med resultatet.

NodeJS

- Open-source, cross-platform JavaScript run-time-miljö som exekverar JavaScript utanför browseren.
- Låter utvecklare använda JS för att skriva Command Line-verktyg och för skript på server-sidan för att producera dynamiska webbsidor. (Alltså precis som PHP.)
- Event-driven arkitektur.
- Hanterar I/O asynkront.

NodeJS

Asynkront

```
const request = require("request");
request(
  "http://swapi.co/api/starships/10/",
  function(err, response, body) {
    console.log("Sen det här");
    console.log(JSON.parse(body));
  }
);
console.log("Det här kommer skrivas ut först");
```

NodeJS

web sockets

- Traditionella requests är stateless, man gör ett request och avslutar sedan kommunikationen tills nästa request skickas.
- Websockets är en webbt teknik där man öppnar en kanal som inte stängs av direkt när anropet avslutats utan går att använda för tvåvägskommunikation mellan server och webbläsare. Om det är chattmeddelanden eller aktieuppdateringar spelar ingen roll.

NodeJS

npm

- npm är en pakethanterare för NodeJS som gör det lätt att ladda hem funktionalitet som andra redan byggt.
- Det är också enkelt att ladda upp sina egna paket.
- Just nu finns det ungefär 430 000 paket som man kan använda sig av.
- Känner ni igen något liknande från PHP-världen?
- Composer

NodeJS

package.json

```
1  {  
2    "name": "realtime-chat",  
3    "scripts": {  
4      "start": "node index"  
5    },  
6    "dependencies": {  
7      "express": "latest",  
8      "socket.io": "^2.1.1"  
9    }  
10 }  
11
```

NodeJS

- Extra uppgift för den som har mycket fritid och vill få mer inblick i hur NodeJS fungerar: bygg en realtids-chat.
- <https://zeit.co/docs/v1/examples/chat>

Webpack

- Huvudsyftet är att paketera JS-filer för användning i en browser.
- Kan även transformera, slå ihop eller paketera andra resurser.
- För den som vill prova:
 - <https://webpack.js.org/guides/installation/>
 - <https://webpack.js.org/guides/getting-started/>

Continuous Integration

- Idén är att göra det lättare att lägga till ny kod genom att automatisera alla kontroller så som:
 - byggstegen
 - applicera lint-regler
 - köra igenom unit-tester
- https://en.wikipedia.org/wiki/Continuous_integration

CircleCI

- CircleCI är gratis att använda för publika GitHub-repon. Den som vill prova kan följa nedanstående guide.
- <https://circleci.com/docs/2.0/getting-started/>

Nyheter i ES6

Constants Constants

Support for constants (also known as "immutable variables"), i.e., variables which cannot be re-assigned new content. Notice: this only makes the variable itself immutable, not its assigned content (for instance, in case the content is an object, this means the object itself can still be altered).

ECMAScript 6 — syntactic sugar: [reduced](#) | [traditional](#)

```
const PI = 3.141593  
PI > 3.0
```

ECMAScript 5 — syntactic sugar: [reduced](#) | [traditional](#)

```
// only in ES5 through the help of object properties  
// and only in global context and not in a block scope  
Object.defineProperty(typeof global === "object" ? global : window,  
  value: 3.141593,  
  enumerable: true,  
  writable: false,  
  configurable: false  
})  
PI > 3.0;
```

Nyheter i ES6

Extended Parameter Handling Default Parameter Values

Simple and intuitive default values for function parameters.

ECMAScript 6 — syntactic sugar: [reduced](#) | [traditional](#)

```
function f (x, y = 7, z = 42) {  
    return x + y + z  
}  
f(1) === 50
```



ECMAScript 5 — syntactic sugar: [reduced](#) | [traditional](#)

```
function f (x, y, z) {  
    if (y === undefined)  
        y = 7;  
    if (z === undefined)  
        z = 42;  
    return x + y + z;  
};  
f(1) === 50;
```



Nyheter i ES6

Extended Parameter Handling Rest Parameter

Aggregation of remaining arguments into single parameter of variadic functions.

ECMAScript 6 — syntactic sugar: [reduced](#) | [traditional](#)

```
function f (x, y, ...a) {  
    return (x + y) * a.length  
}  
f(1, 2, "hello", true, 7) === 9
```

ECMAScript 5 — syntactic sugar: [reduced](#) | [traditional](#)

```
function f (x, y) {  
    var a = Array.prototype.slice.call(arguments, 2);  
    return (x + y) * a.length;  
};  
f(1, 2, "hello", true, 7) === 9;
```

Nyheter i ES6

Classes

Class Definition

More intuitive, OOP-style and boilerplate-free classes.

ECMAScript 6 — syntactic sugar: [reduced](#) | [traditional](#)

```
class Shape {  
  constructor (id, x, y) {  
    this.id = id  
    this.move(x, y)  
  }  
  move (x, y) {  
    this.x = x  
    this.y = y  
  }  
}
```

ECMAScript 5 — syntactic sugar: [reduced](#) | [traditional](#)

```
var Shape = function (id, x, y) {  
  this.id = id;  
  this.move(x, y);  
};  
Shape.prototype.move = function (x, y) {  
  this.x = x;  
  this.y = y;  
};
```

Nyheter i ES6

Classes

Class Inheritance

More intuitive, OOP-style and boilerplate-free inheritance.

ECMAScript 6 — syntactic sugar: [reduced](#) | [traditional](#)

```
class Rectangle extends Shape {
  constructor(id, x, y, width, height) {
    super(id, x, y)
    this.width = width
    this.height = height
  }
}
class Circle extends Shape {
  constructor(id, x, y, radius) {
    super(id, x, y)
    this.radius = radius
  }
}
```

ECMAScript 5 — syntactic sugar: [reduced](#) | [traditional](#)

```
var Rectangle = function(id, x, y, width, height) {
  Shape.call(this, id, x, y);
  this.width = width;
  this.height = height;
};
Rectangle.prototype = Object.create(Shape.prototype);
Rectangle.prototype.constructor = Rectangle;
var Circle = function(id, x, y, radius) {
  Shape.call(this, id, x, y);
  this.radius = radius;
};
Circle.prototype = Object.create(Shape.prototype);
Circle.prototype.constructor = Circle;
```

Nyheter i ES6

Internationalization & Localization Date/Time Formatting

Format date/time with localized ordering and separators.

ECMAScript 6 — syntactic sugar: reduced | traditional

```
var l10nEN = new Intl.DateTimeFormat("en-US")
var l10nDE = new Intl.DateTimeFormat("de-DE")
l10nEN.format(new Date("2015-01-02")) === "1/2/2015"
l10nDE.format(new Date("2015-01-02")) === "2.1.2015"
```

ECMAScript 5 — syntactic sugar: reduced | traditional

```
// no equivalent in ES5
```

Nyheter i ES6

- <http://es6-features.org/>

Spara data i klienten

- Cookies
- Local storage
- (Web SQL Database)
- IndexedDB
- FileSystem API
- <https://www.html5rocks.com/en/tutorials/offline/storage/>

Cookies

- Skapa
 - `document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC";`
- Läsa
 - `let x = document.cookie;`

Local storage

- `localStorage.setItem('myCat', 'Tom');`
- `var cat = localStorage.getItem('myCat');`
- `localStorage.removeItem('myCat');`
- `localStorage.clear();`
- [https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API/Using the Web Storage API](https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API/Using_the_Web_Storage_API)

Hjälp med responsivitet

- <https://search.google.com/test/mobile-friendly>

Yatzy

- Ni bör ha tillräckliga kunskaper för att göra era spel spelbara.
- Vilka data bör man spara på klientsidan och vilka data bör man spara på serversidan?
- Vems tur är det? Hur många slag har spelaren kvar? Vilka fält kan man spara resultatet med nuvarande tärningsuppsättning i? Håll formuläret uppdaterat. Hur många poäng har var och en? Räkna ut summor och bonusar. Slå tärningar. Utse en vinnare.

Sammanfattning

- Datatyper
- Jämförelseoperatorer
- Funktioner
- DOM
- Array-funktioner
- jQuery
- JSON
- API
- Node.JS
- npm
- webpack
- Continuous Integration

Utvärdering

- Prata i grupper om 2-3 personer i två minuter.
- Vad har varit bra idag?
- Vad skulle kunna förbättras?

Tack för idag!

Mikael Olsson
mikael.olsson@emmio.se
076-174 90 43

