Linnéuniversitetet
Institutionen för medieteknik
Interaktiva medier och webbteknologier

http://www.lnu.se/
Växjö
Kandidatprogram 180 hp

Creative Commons (BY-NC-SA)
Henrik Andersen Ph.M.
henrik.andersen@lnu.se

1ME325 Webbteknik 5, 7,5hp

Föreläsning 3b; Objektorienterad programmering med JavaScript (Del 2)

Innehåll

- Omfångshantering med JavaScript
  - Globalt och lokalt omfång
  - Webbläsares window-objekt
- JsOOP; *valideringsverktyg för objektorienterad JavaScript-programmering*

Syfte

- Ge en bättre förståelse för hur webbläsare hanterar JavaScript-kod; *exempelvis vart information lagras under exekvering*

- Introducera verktyg som kan förenkla utvecklingsprocessen och bidra till en bättre förståelse för JavaScripts omfång

Globalt omfång

- Introducerade frågeställningar?
  - *Vad är ett globalt omfång?*
  - *Vad i ECMAScript representerar det globala omfånget?*
  - *Vad finns det för risker med det globala omfånget?*

Globalt omfång

- Det globala omfånget representerar kärnan under exekveringsprocessen; *den plats dit all sessionsbaserad information lagras*

- Det globala omfånget innefattar bland annat:

  - de funktioner och variabler som skapas då skript exekveras av JavaScript-tolken

  - webbläsarens inbyggda objekt så som APIer

Globalt omfång

- Det globala omfånget beskrivs även som webbläsarens window-objekt; *objektet kan nås via den globala referensen window:*

```
window.onload = function() {
    alert("Content loaded!");
};
```

- Ovanstående programkod exekveras då webbläsaren har initierat samtliga underliggande objekt och anses vara redo för användning

Globalt omfång

- Förväntade frågeställningar:
  - *Innebär inte window.onload att DOMen är färdigställd?*
  - *Måste window.onload aktiveras innan JavaScript-kod kan exekveras?*

Globalt omfång

- Då det globala omfånget utgör kärnan i skriptmiljön, adresseras all ickespecificerad information till window-objektet; *allt som deklareras av användaren blir en naturlig del av window*

```
var name = "Henrik Andersen";
window.name = "Henrik Andersen";
name = "Henrik Andersen";
```

- Samtliga av ovanstående kodexempel gör samma sak; *deklarerar en ny variabel (name) som sparas i webbläsarens window-objekt*

Globalt omfång

- Nedanstående kodexempel är en omskrivning av tidigare exempel. Då allt per automatik adresseras till window-objektet, kan programkoden skrivas om till följande:

```
onload = function() {
    alert("Content loaded!");
};
```


Globalt omfång

- Förväntade frågeställningar:
  - *Finns det något rätt, motsvarande fel sätt att deklarera information på?*
  - *Kan man som utvecklare strunta i att använda window-prefixet helt och hållet?*

Globalt omfång; *teoretisk övning*

- Kommande diabild innehåller JavaScript-kod, övningen går ut på att:
  - Identifiera omfång
  - Identifiera eventuella läckage, dvs information som laddats in i det globala window-objektet

Globalt omfång; *teoretisk övning*

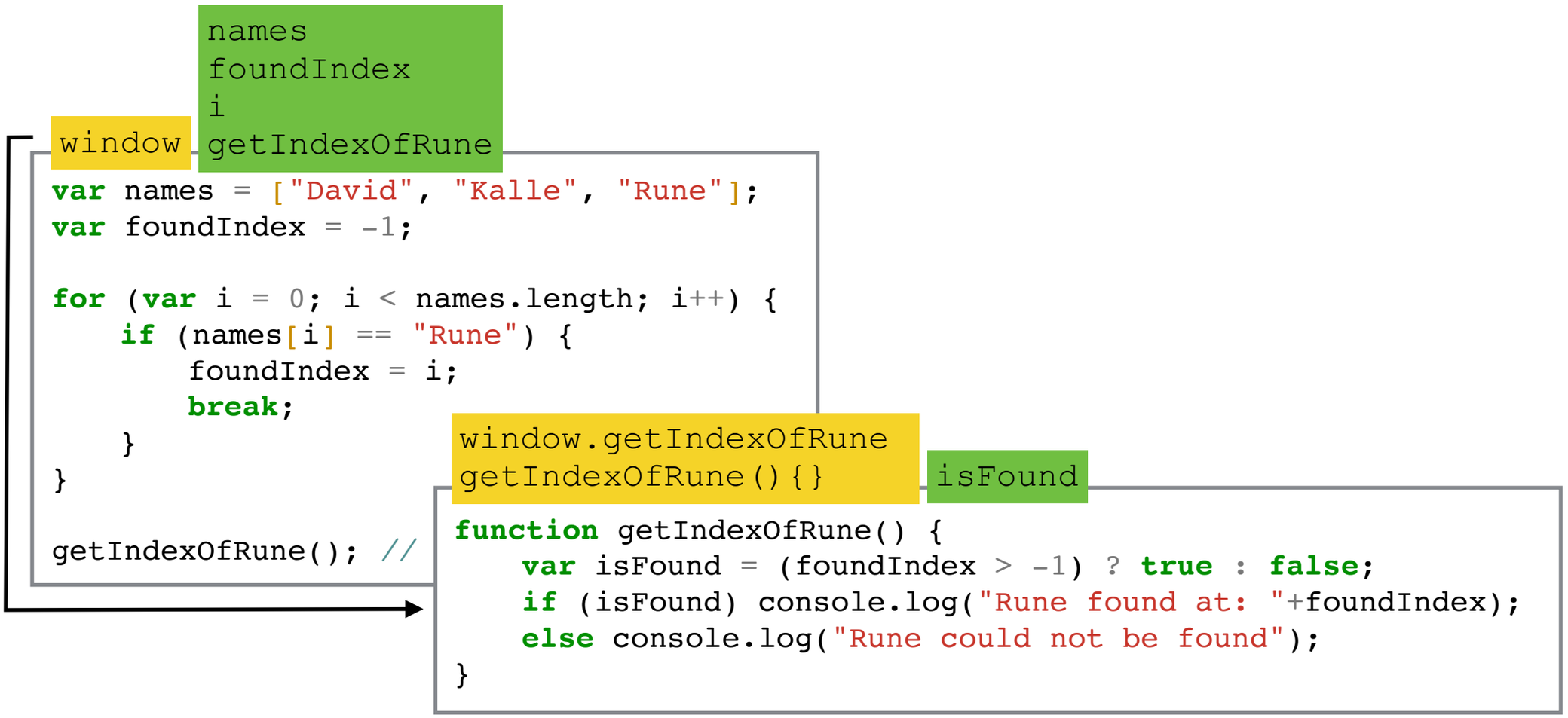
```
var names = ["David", "Kalle", "Rune"];
var foundIndex = -1;

for (var i = 0; i < names.length; i++) {
    if (names[i] == "Rune") {
        foundIndex = i;
        break;
    }
}

function getIndexOfRune() {
    var isFound = (foundIndex > -1) ? true : false;
    if (isFound) console.log("Rune found at: "+foundIndex);
    else console.log("Rune could not be found");
}

getIndexOfRune(); // "Rune found at: 2"
```

Globalt omfång; *teoretisk övning*



Figur. Beskriver omfång i föregående programkod

Globalt omfång; *teoretisk övning*

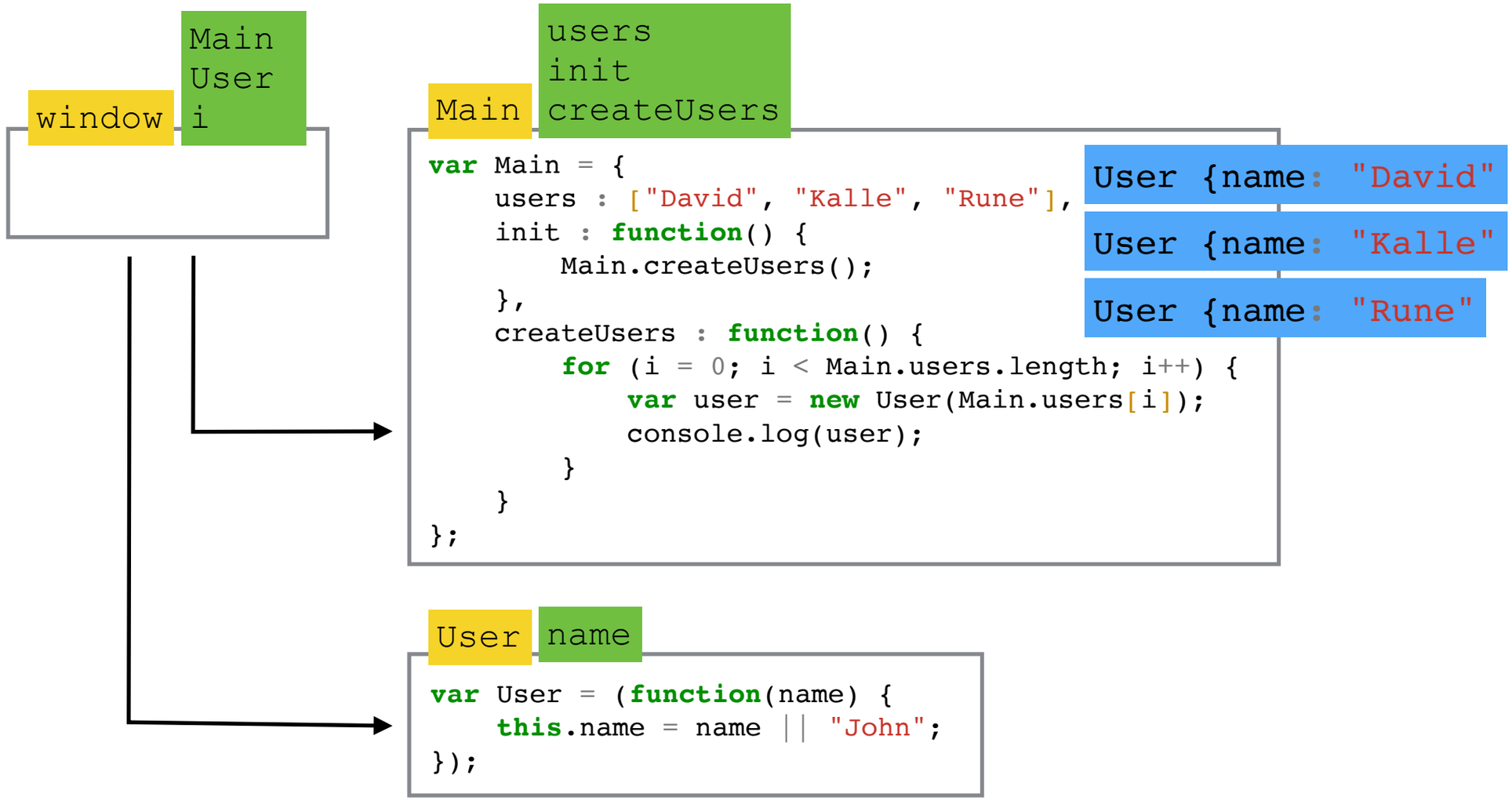
- Förväntade frågeställningar:
  - *Har inte for-loopar ett eget omfång?*
  - *Hur fungerar omfång med object?*

Globalt omfång; *teoretisk övning*

```
var Main = {
    users : ["David", "Kalle", "Rune"],

    init : function() {
        Main.createUsers();
    },
    createUsers : function() {
        for (i = 0; i < Main.users.length; i++) {
            var user = new User(Main.users[i]);
            console.log(user);
        }
    }
};
var User = (function(name) {
    this.name = name || "John";
});
Main.init();
```


Globalt omfång; *teoretisk övning*



Figur. Beskriver omfång i föregående programkod

Globalt omfång; *teoretisk övning*

- Förväntade frågeställningar:
  - *Hur ser åtkomstmöjligheterna ut? kan instanserna nå varandra?*
  - *Hur kan man hålla koll på all?*

Lokala omfång; *kort*

- Ett lokalt omfång är en undergrupp till ett annat lokalt eller globalt omfång

- När information efterfrågas i ett omfång sker följande steg:
  - Leta efter informationen i det egna omfånget med tillhörande prototyp
  - Leta efter informationen i det globala omfånget (window)
  - Om inget påträffas är resultatet `undefined`

Window

- Timers
- Browser location & navigation
- Browsing history
- Browser & screen information
- Dialog boxes
- Error handling
- Document elements as window properties
- Multiple windows and frames

Shepherd, E., et al. (2016). *Window - Web APIs*. Tillgänglig: <https://developer.mozilla.org/en-US/docs/Web/API/Window> [2016-10-28]

Window

- Förväntade frågeställningar:
  - *Kan min programkod påverka det fördefinerade innehållet i window-objektet?*
  - *Kan annan information så som insticksprogram och liknande populera window-objektet?*

JsOOP

- Ett valideringsverktyg för objektorienterad JavaScript med tre primära syften:

  - Belysa hur webbläsare väljer att lagra information; *i vilket omfång som information lagras*

  - Minska risken för felaktig adressering; *att information lagras i det globala omfånget och riskerar att överskriva annan information*

  - Uppmana till objektorienterad programmeringsstruktur; *arbeta med ”klassdokument” och externa filer*

JsOOP; *syfte*

- Verktyget skapades av Andersen och Johansson 2013 som en del av kursen 1ME205: Webbprogrammering 15 hp

- Syftet var att skapa ett hjälpmedel som informerar då information skapas utan förbestämd destination; *detta då många menar att window-objektet i den mån det går, skall förbli orört*

- Är byggt kring en tes och menat att användas i undervisningssyfte

JsOOP; *logik*

- JsOOP följer en förhållandevis enkel logik:
    - Kontrollerar script-element
        - Programkod skall förekomma i externa JavaScript-dokument, inte som inline-kod
        - Externa JavaScript-dokument skall representera klassfiler och därför följa samma regelverk; inledande versal och återkommande namngivning
    - Övervakar det globala omfånget
        - Kontrollerar huruvida det sker förändringar i det globala omfånget
        - Enbart klassdokument får förekomma i det globala omfånget

JsOOP; *användning*

- Källkoden till JsOOP uppdaterades 2016 i primärt syfte att förenkla användningsprocessen
- JsOOP används på följande sätt:
    - Importera JsOOP först bland HTML-dokuments script-element; *all programkod som förekommer efter JsOOP valideras*

JsOOP; *fel & brister*

- JsOOP uppfattar fel och brister i fyra kategorier:

  - Notice; *mindre brister som inte nödvändigtvis inte innebär fel men avvikelser från standardiserad rekommendation*

  - Warning; *brister som kan resultera i fel under exekveringsprocessen eller avvikelser som inte ses som god praxis*

  - Error; *fel som uppstår på grund av avvikelser från objektorienterade riktlinjer, vanligtvis läckage till det globala omfånget*

  - Critical; *regelrätta syntax- eller språkfel som äventyrar verktygets arbetsprocess och därmed trovärdighet*

JsOOP

- Förväntad frågeställning:
  - *Måste man validera sin programkod med JsOOP och i sådana fall varför?*

JsOOP

- Demonstration av JsOOP:
  - https://cactuar.lnu.se/course/1me325/example/jsoop/build/

JsOOP; *sammanfattning*

- Valideringsverktyg som utvecklats i undervisningssyfte
- Görs tillgängligt via kursen och skall användas för att validera den programkod som redogörs i kursuppgiften

Sammanfattning

- JavaScript-utvecklare kan adressera sin programkod till det globala omfånget, eller kapsla in sin programkod i lokala omfång

- Webbläsarens window-objekt representerar det globala omfånget men innefattar även fördefinerade objekt och APIer

- JsOOP är ett valideringsverktyg som följer den arbetsmetodik som kursen förespråkar; *använd verktyget för att validera er programkod*

Frågor