Javascript Sprachfeatures
Scoping, Hoisting, String und Structure

https://iuk.one/1066-1110

Clemens H. **Cap**
ORCID: 0000-0003-3958-6136

Department of Computer Science
University of **Rostock**
Rostock,Germany
clemens.cap@uni-rostock.de

Version 1

1. Scoping

Unterschiede der `var` und `let` scopes.

1. Scoping
2. Hoisting
3. Strukturelle Notation
4. String Features
5. Const

var

```
1  var callbacks = [];
2  for (var i = 0; i <= 2; i++) {
3    callbacks[i] = function() { return i * 2; };
4  }
5
6  // Hat folgende Post-Conditions:
7
8  callbacks[0]() === 6;
9  callbacks[1]() === 6;
10 callbacks[2]() === 6;
```

Src. 1: Var Scoping: Nächst gelegene Funktion oder global.

let

```
1 var callbacks = []
2 for (let i = 0; i <= 2; i++) {
3     callbacks[i] = function () { return i * 2 }
4 }
5
6 callbacks[0]() === 0
7 callbacks[1]() === 2
8 callbacks[2]() === 4
```

Src. 2: Let Scoping: Nächst gelegener Block.

Closures

```
1  var callbacks = [];
2  for (var i = 0; i <= 2; i++) {
3      (function (i) {  // closure !
4          callbacks[i] = function() { return i * 2; };
5      })(i);
6  }
7
8  callbacks[0]() === 0;
9  callbacks[1]() === 2;
10 callbacks[2]() === 4;
```

Src. 3: Closure Scoping

2. Hoisting

Hochheben der Definitionszeitpunkte

1. Scoping
2. Hoisting
3. Strukturelle Notation
4. String Features
5. Const

Deklarationen und Hoisting

var deklarierte Variable

Deklarationen lokaler Variable mit var werden an den Anfang des scope gehoben (Hoisting).

let deklarierte Variable

Ist erst ab dem let deklariert
Liefert davor Referenzfehler.

Nicht deklarierte Variable

Wird dem obersten Objekt (windows) zugewiesen.

Beispiel: Undeklarierte Variable

```
1 function foo() {
2   x=3;
3   console.log (x);         // 3
4   console.log (window.x); // 3
5 }
6 foo();
```

Src. 4: Nicht deklarierte Variable wird dem obersten Objekt zugewiesen.

Beispiel: Hoisting

```
1 function foo() {
2   x=3;
3   console.log (x);          // 3
4   console.log (window.x); // undefined
5   var x;
6 }
7 foo();
```

Src. 5: Hoisting hebt den Deklarationszeitpunkt zum Beginn der aktuellen Funktion.

Beispiel: Verspätet let deklarierte Variable

```
1 function foo() {
2   x=3;                        // reference error
3   console.log (x);
4   console.log (window.x);
5   let x;                      // bei let passiert kein hoisting
6 }
7 foo();
```

Src. 6: Verspätet let deklarierte Variable

3. Strukturelle Notation

Syntaktische Erweiterungen
für komplexere Strukturen

1. Scoping
2. Hoisting
3. Strukturelle Notation
4. String Features
5. Const

Kurznotationen für Objekte

```
1  obj = { x, y };                                    // als Kurznotation für
2  obj = { x: x, y: y };
3
4  obj = { foo: "bar", [ "baz" + more() ]: 42 };      // als Kurznotation für
5  obj = { foo: "bar" }; obj[ "baz" + more() ] = 42;
6
7  obj = { foo (a,b) {...}, bar (x,y) {...}, *q (x,y) {...} }  // als Kurznotation für
8  obj = { foo: function (a, b) { ... }, bar: function (x, y) { ... }, };
9  //*q ohne Entsprechung; Generatorfunktion
```

Src. 7: Kurznotationen für Objekte

Destrukturieren einer Liste

Besonders hilfreich beim Destrukturieren von Rückgabewerten einer Funktion

```
1  var list = [ 1, 2, 3 ]; var a = list[0], b = list[2];  // alt
2  var list = [ 1, 2, 3 ]; var [ a, , b ] = list;         // neu
3
4  const cells = 'Jane\tDoe\tCTO'
5  const [firstName, lastName, title] = cells.split('\t');
6
7  // Funktioniert bei allen Iterable Values:
8  const [x,y] = new Set().add('a').add('b');  // x = 'a'; y = 'b'
9  const [a,b] = 'foo';                        // a = 'f'; b = 'o'
```

Src. 8: Destrukturieren einer Liste

Swapping Notation

```
1 var tmp = a; a = b; b = tmp;
```

Src. 9: Swapping bisher

```
1 [ b, a ] = [ a, b ]
```

Src. 10: Neue Swapping Notation

Array Matching Beispiele

```
1  // Angewendet als Teil einer for...of Iteration
2  const map = new Map().set(false, 'no').set(true, 'yes');
3  for (const [key, value] of map)
4    {console.log(key + ' is ' + value);}
5
6  // In Kombination mit dem Spread Operator
7  const [first, ...rest] = ['a', 'b', 'c'];
8  // first = 'a'; rest = ['b', 'c'];
```

Src. 11:

Grundsätzlich:

- Das Meiste ist intuitiv verständlich
- Einige wenige Pitfalls

Rückgabewerte

```
1  var { op, lhs, rhs } = getASTNode();     // neu
2
3  var tmp = getASTNode();                  // bisher
4  var op  = tmp.op;                        // bisher
5  var lhs = tmp.lhs;                       // bisher
6  var rhs = tmp.rhs;                       // bisher
```

Src. 12: Strukturelle Notation für Rückgabewerte

Sonderfall

```
1  { a, b } = someObject;          // SyntaxError
2  ({ a, b } = someObject);        // OK
3
4  ({ a, b }) = someObject;        // SyntaxError
5  const { a, b } = someObject;    // OK
```

Src. 13: Sonderfall

Achtung: Fehlen `var` oder `let`, dann braucht es runde `()` Klammern da geschwungene Klammern für Code-Blöcke stehen.

Im Funktionsaufruf

```
1  // Neu
2  function f ([ name, val ])   {console.log(name, val)}
3  function g ({ name: n, val: v })  {console.log(n, v)}
4  function h ({ name, val })   {console.log(name, val)}
5  f([ "bar", 42 ])
6  g({ name: "foo", val:  7 })
7  h({ name: "bar", val: 42 })
8
9  // Bisher
10 function f (arg) {var name = arg[0]; var val  = arg[1]; ...};
11 function g (arg) {var n = arg.name; var v = arg.val; ... };
12 function h (arg) {var name = arg.name; var val  = arg.val; ...};
13 f([ "bar", 42 ]);
14 g({ name: "foo", val:  7 });
15 h({ name: "bar", val: 42 });
```

Src. 14: Strukturelle Notation im Funktionsaufruf

Weitere Beispiele mit Listen

```
1  // Neu
2  var list = [ 7, 42 ];
3  var [ a = 1, b = 2, c = 3, d ] = list;
4  a === 7
5  b === 42
6  c === 3
7  d === undefined
8
9  // Bisher
10 var list = [ 7, 42 ];
11 var a = typeof list[0] !== "undefined" ? list[0] : 1;
12 var b = typeof list[1] !== "undefined" ? list[1] : 2;
13 var c = typeof list[2] !== "undefined" ? list[2] : 3;
14 var d = typeof list[3] !== "undefined" ? list[3] : undefined;
15 a === 7; b === 42; c === 3; d === undefined;
```

Src. 15: Weitere Beispiele mit Listen

4. String Features

Weitere Features

1. Scoping
2. Hoisting
3. Strukturelle Notation
4. String Features
5. Const

Warnung

Beachte:

- Fehler in String-Verarbeitung sind schwer zu erkennen.
- Hängen oft von feinen syntaktischen Details ab Bsp: Unterscheide: ´ ‘ ’ "
- Benötigen oftmals trickreiches Escapement.
- Fehler in String-Verarbeitung: Nummer 1 Ursache für injection Attacken
- Injection Attacken: Nummer 1 Ursache für Datendiebstahl am Server

```
1 var txt = '<a title="Das \" ist ein doppeltes Hochkomma" href="' + url + '">';
```

Src. 16: Trickreiches Escapement

String Interpolation

Thema: Einbauen von Daten in Strings
Typisch: Durch Template Engines

Probleme:

- Viele verschiedene Stufen von Quotes
- Quote Escapements
- Viele String Additionsoperatoren
- Daher schwer lesbar.

Beispiel für Interpolation

```
1 var customer = { name: "Foo" };
2 var card = { amount: 7, product: "Bar", unitprice: 42 }
3 message = `Hello ${customer.name},
4 want to buy ${card.amount} ${card.product} for
5 a total of ${card.amount * card.unitprice} bucks?`
```

Src. 17: String Interpolation

Beachte das besondere, "schiefe" statt "gerade" einzelne Hochkomma (gravis)

Multiline Text

```
Multiline Text
const multiLine = `
This is
a string
with multiple
lines`;
```

Src. 18: Multiline Text

Internationalisierung: Collatierung

Collatierung = Sortierreihenfolge

Beispiel:

1. Deutsch: ä sortiert zusammen mit a
2. Schwedisch: ä sortiert nach dem z

```
1 var list = [ "ä", "a", "z" ]
2 var l10nDE = new Intl.Collator("de")
3 var l10nSV = new Intl.Collator("sv")
4 l10nDE.compare("ä", "z") === -1
5 l10nSV.compare("ä", "z") === +1
6 console.log(list.sort(l10nDE.compare)) // [ "a", "ä", "z" ]
7 console.log(list.sort(l10nSV.compare)) // [ "a", "z", "ä" ]
```

Src. 19: Sortierreihenfolge

Internationalisierung: Zahlenformatierung

Wie erfolgt die Gruppierung und Separierung von Zahlen?

```
1 var l10nEN = new Intl.NumberFormat("en-US")
2 var l10nDE = new Intl.NumberFormat("de-DE")
3 l10nEN.format(1234567.89) === "1,234,567.89"
4 l10nDE.format(1234567.89) === "1.234.567,89"
```

Src. 20: Internationalisierung: Zahlenformatierung

Internationalisierung: Währungsformatierung

```
1 var l10nUSD = new Intl.NumberFormat("en-US", { style: "currency", currency: "USD" })
2 var l10nGBP = new Intl.NumberFormat("en-GB", { style: "currency", currency: "GBP" })
3 var l10nEUR = new Intl.NumberFormat("de-DE", { style: "currency", currency: "EUR" })
4 l10nUSD.format(100200300.40) === "\$100,200,300.40"
5 l10nGBP.format(100200300.40) === "£100,200,300.40"
6 l10nEUR.format(100200300.40) === "100.200.300,40 €"
```

Src. 21: Internationalisierung: Währungsformatierung

Internationalisierung: Zeit und Datum

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

Src. 22: Internationalisierung: Zeit und Datum

5. Const

1. Scoping
2. Hoisting
3. Strukturelle Notation
4. String Features
5. Const

Konstante

- Variablen können als konstant deklariert werden.
- Bei primitive Typen: Wert nicht änderbar
- Bei Referenzen: Referenz nicht änderbar (referenziertes Objekt schon!)

```
1 const PI = 3.141593
```

Src. 23: Konstante

Anhang

Übersicht

Übersicht

Programmquellenverzeichnis Prog

Rechtliche Hinweise §

Zitierweise dieses Dokuments →

Verzeichnis aller Folien

Programmquellenverzeichnis (1/3)

1 Var Scoping: Nächst gelegene Funktion oder global. 3
2 Let Scoping: Nächst gelegener Block. 4
3 Closure Scoping 5
4 Nicht deklarierte Variable wird dem obersten Objekt zugewiesen. 8
5 Hoisting hebt den Deklarationszeitpunkt zum Beginn der aktuellen Funktion. 9
6 Verspätet let deklarierte Variable 10
7 Kurznotationen für Objekte 12
8 Destrukturieren einer Liste 13
9 Swapping bisher 14
10 Neue Swapping Notation 14

11 15
12 Strukturelle Notation für Rückgabewerte 16
13 Sonderfall 17
14 Strukturelle Notation im Funktionsaufruf 18
15 Weitere Beispiele mit Listen 19
16 Trickreiches Escapement 21
17 String Interpolation 23
18 Multiline Text 24
19 Sortierreihenfolge 25
20 Internationalisierung: Zahlenformatierung 26
21 Internationalisierung: Währungsformatierung 27

22 Internationalisierung: Zeit und Datum 28
23 Konstante 30

Rechtliche Hinweise (1)

Die hier angebotenen Inhalte unterliegen deutschem Urheberrecht. Inhalte Dritter werden unter Nennung der Rechtsgrundlage ihrer Nutzung und der geltenden Lizenzbestimmungen hier angeführt. Auf das Literaturverzeichnis wird verwiesen. Das Zitatrecht in dem für wissenschaftliche Werke üblichen Ausmaß wird beansprucht. Wenn Sie eine Urheberrechtsverletzung erkennen, so bitten wir um Hinweis an den auf der Titelseite genannten Autor und werden entsprechende Inhalte sofort entfernen oder fehlende Rechtsnennungen nachholen. Bei Produkt- und Firmennamen können Markenrechte Dritter bestehen. Verweise und Verlinkungen wurden zum Zeitpunkt des Setzens der Verweise überprüft; sie dienen der Information des Lesers. Der Autor macht sich die Inhalte, auch in der Form, wie sie zum Zeitpunkt des Setzens des Verweises vorlagen, nicht zu eigen und kann diese nicht laufend auf Veränderungen überprüfen.

Alle sonstigen, hier nicht angeführten Inhalte unterliegen dem Copyright des Autors, Prof. Dr. Clemens Cap, ©2020. Wenn Sie diese Inhalte nützlich finden, können Sie darauf verlinken oder sie zitieren. Jede weitere Verbreitung, Speicherung, Vervielfältigung oder sonstige Verwertung außerhalb der Grenzen des Urheberrechts bedarf der schriftlichen Zustimmung des Rechteinhabers. Dieses dient der Sicherung der Aktualität der Inhalte und soll dem Autor auch die Einhaltung urheberrechtlicher Einschränkungen wie beispielsweise Par 60a UrhG ermöglichen.

Die Bereitstellung der Inhalte erfolgt hier zur persönlichen Information des Lesers. Eine Haftung für mittelbare oder unmittelbare Schäden wird im maximal rechtlich zulässigen Ausmaß ausgeschlossen, mit Ausnahme von Vorsatz und grober Fahrlässigkeit. Eine Garantie für den Fortbestand dieses Informationsangebots wird nicht gegeben.

Die Anfertigung einer persönlichen Sicherungskopie für die private, nicht gewerbliche und nicht öffentliche Nutzung ist zulässig, sofern sie nicht von einer offensichtlich rechtswidrig hergestellten oder zugänglich gemachten Vorlage stammt.

Use of Logos and Trademark Symbols: The logos and trademark symbols used here are the property of their respective owners. The YouTube logo is used according to brand request 2-9753000030769 granted on November 30, 2020. The GitHub logo is property of GitHub Inc. and is used in accordance to the GitHub logo usage conditions https://github.com/logos to link to a GitHub account. The Tweedback logo is property of Tweedback GmbH and here is used in accordance to a cooperation contract.

Rechtliche Hinweise (2)

Disclaimer: Die sich immer wieder ändernde Rechtslage für digitale Urheberrechte erzeugt ein nicht unerhebliches Risiko bei der Einbindung von Materialien, deren Status nicht oder nur mit unverhältnismäßig hohem Aufwand abzuklären ist. Ebenso kann den Rechteinhabern nicht auf sinnvolle oder einfache Weise ein Honorar zukommen, obwohl deren Leistungen genutzt werden.

Daher binde ich gelegentlich Inhalte nur als Link und nicht durch Framing ein. Lt EuGH Urteil 13.02.2014, C-466/12 (Pressemitteilung, Blog-Beitrag, Urteilstext). ist das unbedenklich, da die benutzten Links ohne Umgehung technischer Sperren auf im Internet frei verfügbare Inhalte verweisen.

Wenn Sie diese Rechtslage stört, dann setzen Sie sich für eine Modernisierung des völlig veralteten Vergütungs- und Anreizsystems für urheberrechtliche Leistungen ein. Bis dahin klicken Sie bitte auf die angegebenen Links und denken Sie darüber nach, warum wir keine für das digitale Zeitalter sinnvoll angepaßte Vergütungs- und Anreizsysteme digital erbrachter Leistungen haben.

Zu Risiken und Nebenwirkungen fragen Sie Ihren Rechtsanwalt oder Gesetzgeber.

Weitere Hinweise finden Sie im Netz hier und hier oder hier.

Zitierweise dieses Dokuments

Wenn Sie Inhalte aus diesem Werk nutzen oder darauf verweisen wollen,
zitieren Sie es bitte wie folgt:

Clemens H. Cap: Javascript Sprachfeatures
Scoping, Hoisting, String und Structure. Electronic document. https://iuk.one/1066-1110 18. 6. 2023.

Bibtex Information: https://iuk.one/1066-1110.bib

```
@misc{doc:1066-1110,
   author       = {Clemens H. Cap},
   title        = {Javascript Sprachfeatures
Scoping, Hoisting, String und Structure},
   year         = {2023},
   month        = {6},
   howpublished = {Electronic document},
   url          = {https://iuk.one/1066-1110}
}
```

Typographic Information:
Typeset on ?today?
This is pdfTeX, Version 3.14159265-2.6-1.40.21 (TeX Live 2020) kpathsea version 6.3.2
This is pgf in version 3.1.5b
This is preamble-slides.tex myFormat©C.H.Cap

Verzeichnis aller Folien

Titelseite 1

1. Scoping

var 3
let 4
Closures 5

2. Hoisting

Deklarationen und Hoisting 7
Beispiel: Undeklarierte Variable 8
Beispiel: Hoisting 9
Beispiel: Verspätet let deklarierte Variable 10

3. Strukturelle Notation

Kurznotationen für Objekte 12
Destrukturieren einer Liste 13
Swapping Notation 14
Array Matching Beispiele 15
Rückgabewerte 16
Sonderfall 17
Im Funktionsaufruf 18
Weitere Beispiele mit Listen 19

4. String Features

Warnung 21
String Interpolation 22
Beispiel für Interpolation 23
Multiline Text 24
Internationalisierung: Collatierung 25
Internationalisierung: Zahlenformatierung 26
Internationalisierung: Währungsformatierung 27
Internationalisierung: Zeit und Datum 28

5. Const

Konstante 30

Legende:

- Fortsetzungsseite
- Seite ohne Überschrift
- Bildseite