WEBTECHNOLOGIEN

03 – JAVASCRIPT
CONST LET UND OBJEKTE

PROF. DR. MARKUS HECKNER

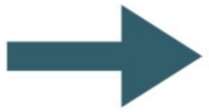
const

- Similar to the var statement*
- However, the value cannot be redeclared or reassigned.
- It is thus CONSTANT

```
// String
const greeting = 'hello';
// Number
const favoriteNum = 33;
// Boolean
const isAwesome = true;
```

** but block scoped. More on this later…*

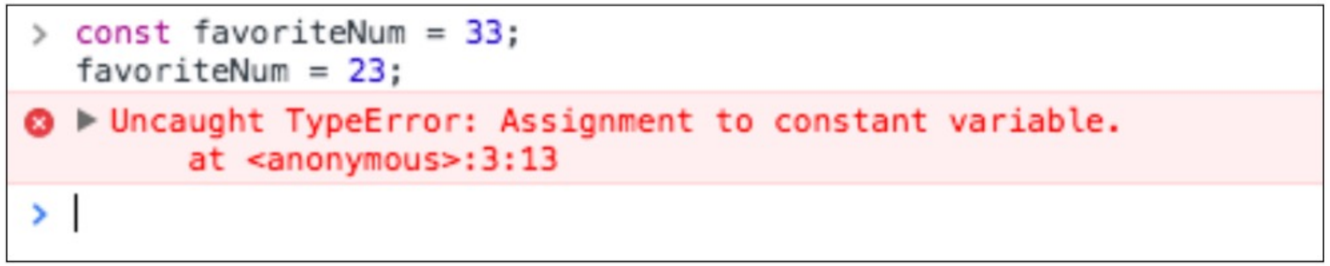
const Errors



```
// Number
const favoriteNum = 33;

favoriteNum = 23;
```

- Cannot change your mind once const initialised
- Reassignment prohibited - error if attempted.



```
> const favoriteNum = 33;
  favoriteNum = 23;
⊗ ▶ Uncaught TypeError: Assignment to constant variable.
        at <anonymous>:3:13
> |
```

VERWENDUNG VON LET ERMÖGLICHT DIE VERWENDUNG VON BLOCKVARIABLEN

- `let` ermöglicht es Variablen zu deklarieren, deren Gültigkeitsbereich auf den **Block**, […] beschränkt ist, in dem sie deklariert sind.

?

Welche Ausgaben erzeugen die beiden Funktionsaufrufe?

```
function varTest() {
  var x = 31;
  if (true) {
    var x = 71;
    console.log(x);
  }
  console.log(x);
}

function letTest() {
  let x = 31;
  if (true) {
    let x = 71;
    console.log(x);
  }
  console.log(x);
}
```

Quelle: https://developer.mozilla.org/de/docs/Web/JavaScript/Reference/Statements/let

JAVASCRIPTBASICS

ES GIBT 5 PRIMITIVES (NUMBER, STRING, BOOLEAN, UNDEFINED UND NULL) UND KOMPLEXE TYPEN

Komplexe Datentypen | Array | Eigene Objekte | Function

- Die Länge eines Arrays entspricht (wie in Java und C) dem höchsten Index + 1
- Achtung: Arrays lassen sich zur Laufzeit dynamisch verlängern! Das geht in Java nicht...

```
var shoppingItems = [];
shoppingItems[0] = "Orangensaft";
shoppingItems[1] = "Tomaten";
shoppingItems[2] = "Pizza";

console.log(shoppingItems.length);
//Outputs: 3
shoppingItems[4] = "Nudeln";
console.log(shoppingItems.length);
//Outputs: 5
```

? Welche Ausgaben erzeugt der Code?

JAVASCRIPTBASICS

ES GIBT 5 PRIMITIVES (NUMBER, STRING, BOOLEAN, UNDEFINED UND NULL) UND KOMPLEXE TYPEN

Komplexe Datentypen | **Array** | Eigene Objekte | Function

- Nicht gesetzte Indizes in einem Array werden als `undefined` zurückgegeben

```
var shoppingItems = [];
shoppingItems[0] = "Orangensaft";
shoppingItems[1] = "Tomaten";
shoppingItems[2] = "Pizza";
shoppingItems[4] = "Nudeln";

for(var i = 0; i< shoppingItems.length; i++){
    console.log(shoppingItems[i]);
}
//Outputs: "Orangensaft", "Tomaten", "Pizza”,
undefined, "Nudeln"
```

? Welche Ausgaben erzeugt der Code?

JAVASCRIPTBASICS
EIGENE OBJEKTE ERSTELLEN MIT OBJEKTLITERAL

Komplexe Datentypen | Array | Eigene Objekte | Function

!

Diesen Weg der Objekterzeugung benötigen wir später (vgl. Module Pattern!)

- JavaScript-Objekte sind einfache Schlüssel-Wert Paare (vgl. Java HashMaps, C Hash Tabellen, Dictionaries in Python)
- Verwendung ähnlich zu `structs` in C: Daten, aber keine Methoden zu den Daten

Erzeugen eines neuen Objekts mit Objektliteral

```
let shoppingListItem = {
    name: "Cola",
    price: 1.99,
    quantity: 10
};
```

Werte

Eigenschaften

```
console.log(shoppingListItem.name);
console.log(shoppingListItem.price);
console.log(shoppingListItem.quantity);
```

?

Welche Ausgaben erzeugt der Code?

Object Creation Object Literal

Vgl. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Basics

Objects with Strings & Numbers

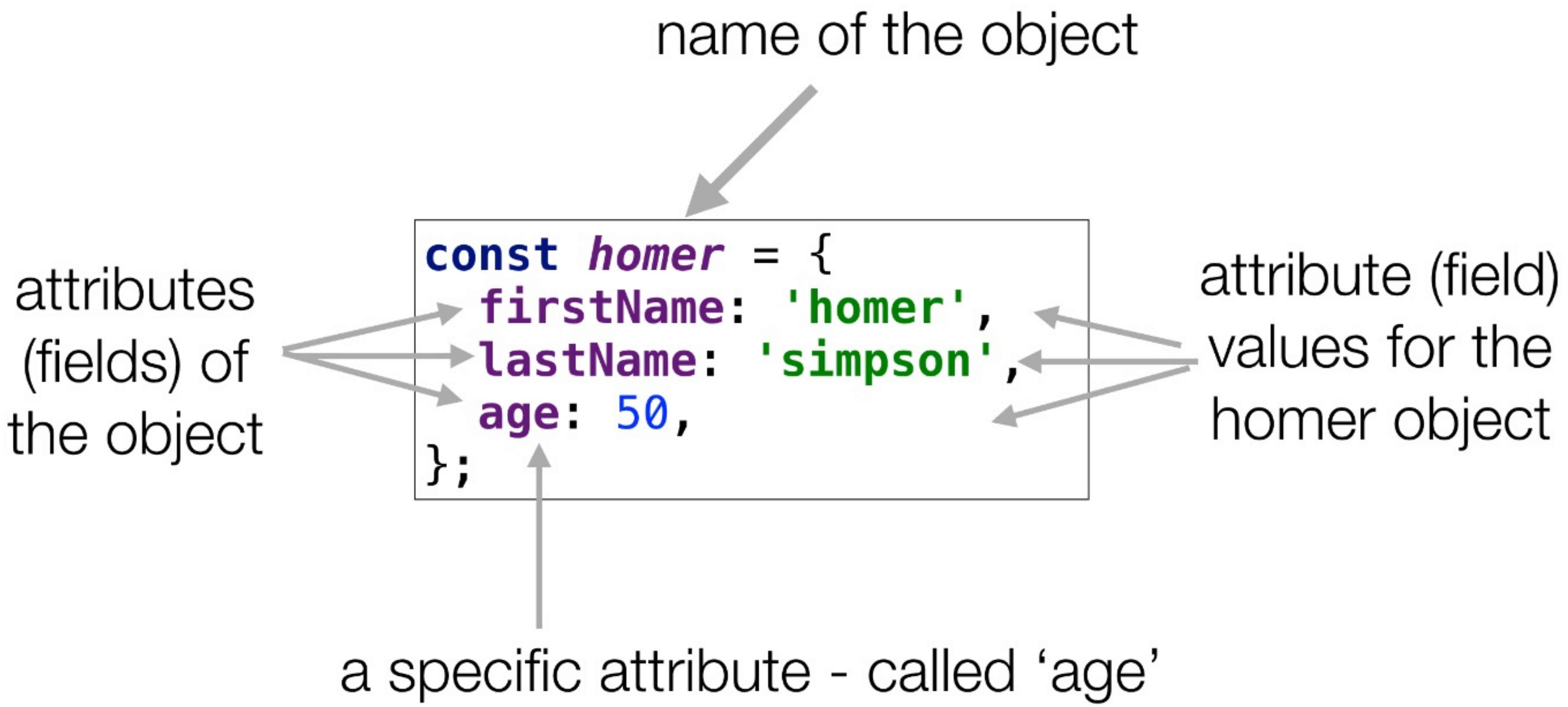
```
const bart = {
  firstName: 'bart',
  lastName: 'simpson',
  age: 10,
};

console.log(bart);
```

- An object containing 2 strings and a number.

```
{ firstName: 'homer', lastName: 'simpson' }
```

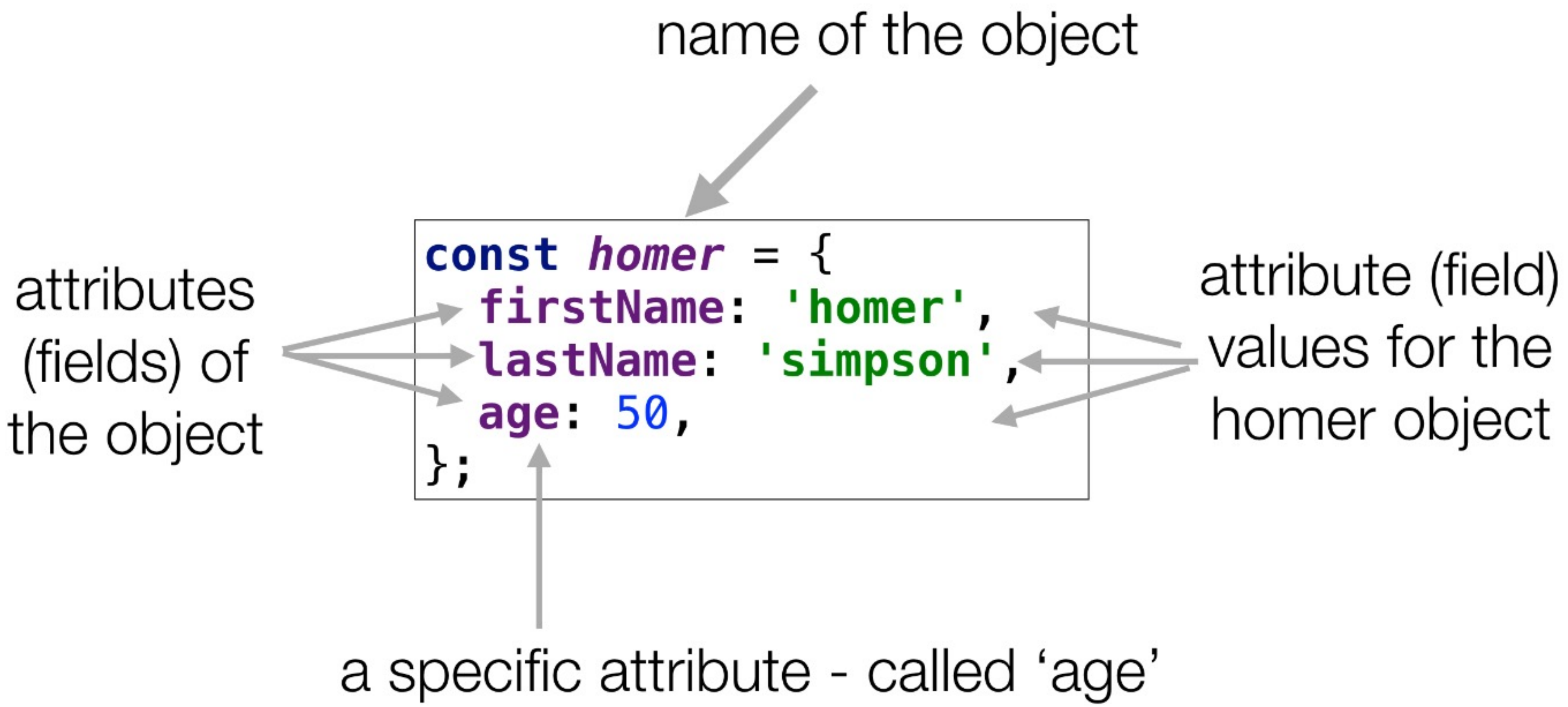
Anatomy of an Object



```
const homer = {
  firstName: 'homer',
  lastName: 'simpson',
  age: 50,
};
```

name of the object
attributes (fields) of the object
attribute (field) values for the homer object
a specific attribute - called 'age'

Anatomy of an Object

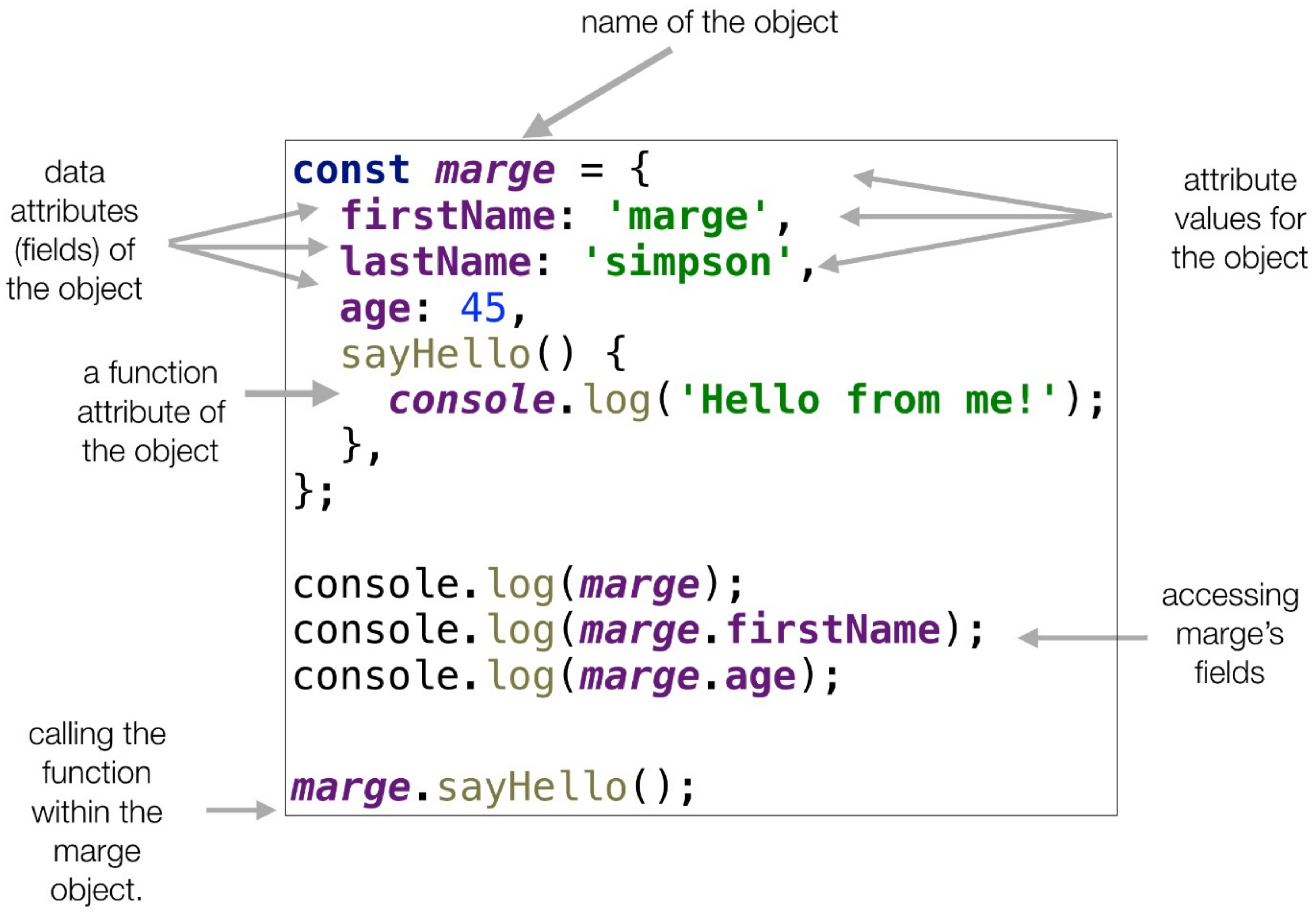

name of the object
attributes (fields) of the object
attribute (field) values for the homer object
a specific attribute - called 'age'

```
const homer = {
  firstName: 'homer',
  lastName: 'simpson',
  age: 50,
};
```

Objects with Functions

```
const marge = {
  firstName: 'marge',
  lastName: 'simpson',
  age: 10,
  sayHello() {
    console.log('Hello from me!');
  },
};

marge.sayHello();
```


name of the object
data attributes (fields) of the object
attribute values for the object
a function attribute of the object
accessing marge's fields
calling the function within the marge object.

```
const marge = {
  firstName: 'marge',
  lastName: 'simpson',
  age: 45,
  sayHello() {
    console.log('Hello from me!');
  },
};

console.log(marge);
console.log(marge.firstName);
console.log(marge.age);


marge.sayHello();
```

this refers to the 'current' object. Ned in this case

```
const ned = {
  firstName: 'ned',
  lastName: 'flanders',
  age: 45,
  speak() {
    console.log('How diddley do? says ' + this.firstName);
  },
};

ned.speak();
```

```
How diddley do? says ned
```

https://slides.com/concise/js/

concise JavaScript

A concise and accurate JavaScript tutorial/notes written for those entering the JavaScript world for the first time but already have experience with other languages

Some slides extracted from above reference

Definition

A method is *a* ***function*** as *some object's property*

The property which contains a value that
references to some function is called a "method."

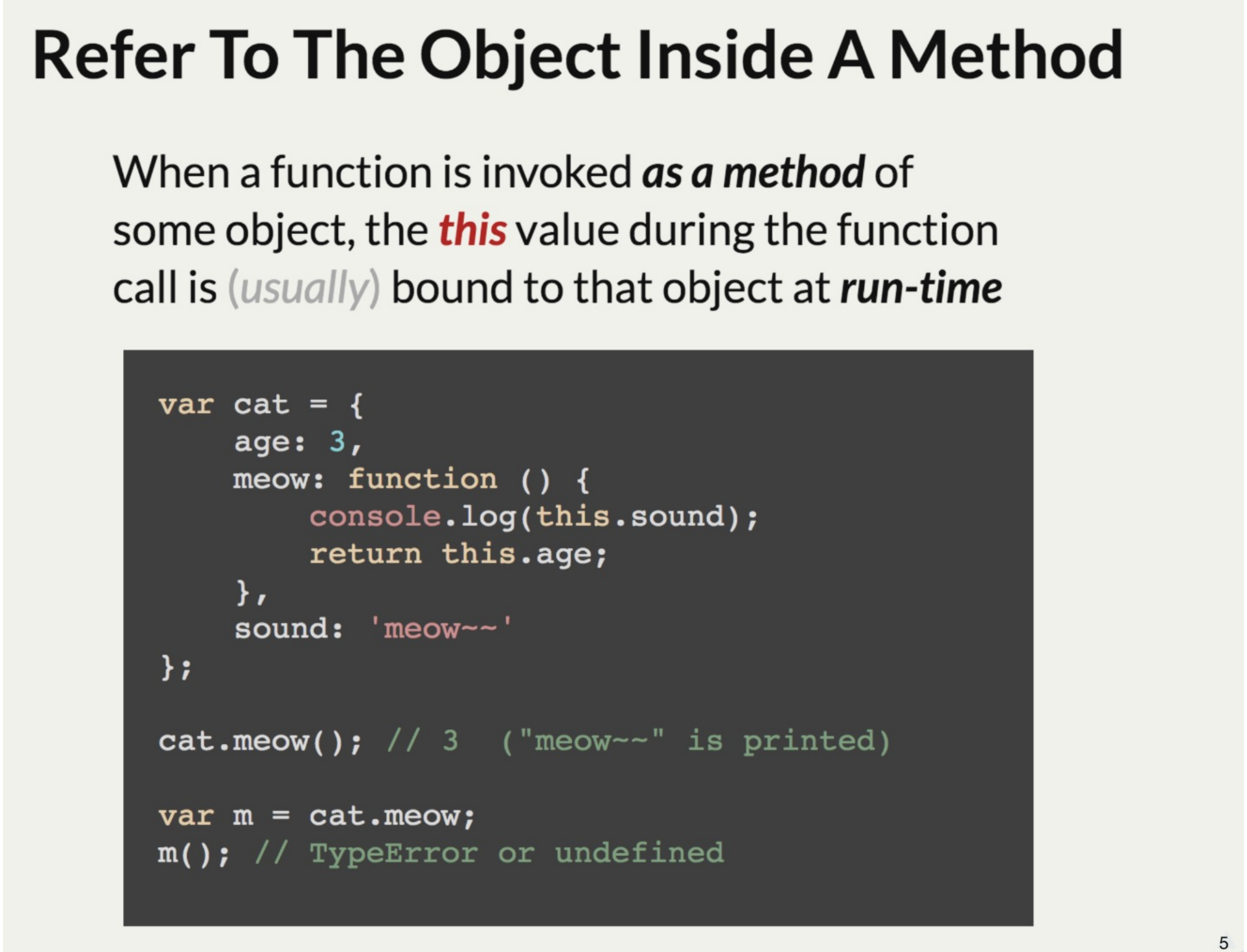
So is the referenced function.

Methods of An Object

```
// The cat object has three properties
// cat.age, cat.meow, and cat.sleep

var cat = {
    age: 3,
    meow: function () {}
};
cat.sleep = function () {};

// We would say that cat.meow and
// cat.sleep are "methods" of cat
```



Refer To The Object Inside A Method

When a function is invoked ***as a method*** of some object, the ***this*** value during the function call is *(usually)* bound to that object at ***run-time***

```
var cat = {
    age: 3,
    meow: function () {
        console.log(this.sound);
        return this.age;
    },
    sound: 'meow~~'
};

cat.meow(); // 3  ("meow~~" is printed)

var m = cat.meow;
m(); // TypeError or undefined
```

5

Methods

```
var cat = {
  age: 3,
  meow : function () {
    console.log(this.sound);
    return this.age;
  },
  sound: 'meow~~'
};

cat.meow();
```

Shorthand syntax for Methods

```
var cat = {
  age: 3,
  meow () {
    console.log(this.sound);
    return this.age;
  },
  sound: 'meow~~'
};

cat.meow();
```

QUELLEN

Imbert, T. (2013). A JavaScript Refresh. Online verfügbar: http://typedarray.org/JavaScript-refresh/. Letzter Zugriff: 11.08.2015.

Mozilla Developer Network. (2015b). A re-introduction to JavaScript (JS tutorial). Online verfügbar: https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript

Mozilla Developer Network. (2015b). Introduction to Object Oriented JavaScript. Online verfügbar: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Introduction_to_Object-Oriented_JavaScript#JavaScript_object_oriented_programming. Letzter Zugriff: 13.08.2015