JavaScript

Ivan Hell

23. Januar 2013

Content

What is JavaScript?

- JavaScript is not Java

What is JavaScript?

- JavaScript is not Java
- Scripting language for manipulation of web-pages

What is JavaScript?

- JavaScript is not Java
- Scripting language for manipulation of web-pages
- JavaScript engines are usually parts of web browsers

What is JavaScript?

- JavaScript is not Java
- Scripting language for manipulation of web-pages
- JavaScript engines are usually parts of web browsers
- Used on many websites (Google, Amazon, . . .)

History

- Developed by Brendan Eich in 1995 under the name LiveScript
- Feature for *Netscape Navigator 2.0* that should be able to verify form inputs
- Renamed to JavaScript after an agreement with *Sun MicroSystems*

Figure : Brendan Eich

Image: http://news.oreilly.com/2008/08/18/BEich.jpg

History

- Rollover-graphics introduced with JavaScript 1.1
- *JScript*, proprietary implementation of JavaScript for the *Internet Explorer 3*
- 1997: ECMA-262 standard released
- 1998: Netscape was bought by AOL $\rightarrow$ Netscape-browser was outsourced as open-source-project *Mozilla*
- Current stable version: JavaScript 1.8.5

What is Javascript used for?

JavaScript is mainly used to:

- Verify form inputs
- Dynamic manipulation of websites (DOM)
- Send and recieve data without reloading the whole website (AJaX)
- Small web-applications (timers, color-choosers, . . .)
- Many Google web-applications (Google maps, Google docs, . . .)

DOM and AJaX

Data **O**bject **M**odel:

HTML-page is repesented as an object with certain properties, where all of this properties can be accessed and changed using JavaScript (for example the object `document.title` can be used to change a websites title).

DOM and AJaX

Data **O**bject **M**odel:

HTML-page is repesented as an object with certain properties, where all of this properties can be accessed and changed using JavaScript (for example the object `document.title` can be used to change a websites title).

Asynchronous **J**avaScript **a**nd **X**ML:

In this use case JavaScript is used to send HTTP requests to a server, while the actual website is displayed. After recieving the answers the displayed page can be modified using this data without reloading the whole page. Data exchange usually was implemented using XML.

Language properties

Dynamic Typing:

Variables are not bound to a specific data-type. Type checks are performed mostly at run time $\rightarrow$ often called *value typing*.

Language properties

Dynamic Typing:

Variables are not bound to a specific data-type. Type checks are performed mostly at run time $\rightarrow$ often called *value typing*.

Weak Typing:

Interpreter is able to automatically cast data of a given type into another type at run time. Allows to use operations for types, for which they were not designed.

Paradigms

Object oriented:

- All data is represented as object
- Objects have properties that are objects too
- Inheritance implemented using prototypes

Paradigms

Object oriented:

- All data is represented as object
- Objects have properties that are objects too
- Inheritance implemented using prototypes

Imperative:

- Script defined as a sequence of commands
- Developer specfies *how* to do something
- Several control-structures

Paradigms

Object oriented:

- All data is represented as object
- Objects have properties that are objects too
- Inheritance implemented using prototypes

Imperative:

- Script defined as a sequence of commands
- Developer specfies *how* to do something
- Several control-structures

Functional:

- Passing functions as argument to other functions
- Each function seen as an individual object
- Not a *pure* functional language

Data types

Number:

- Unique data-types for floating-point and integer numbers
- Internally all represented as 64-bit floating point numbers

Data types

Number:

- Unique data-types for floating-point and integer numbers
- Internally all represented as 64-bit floating point numbers

String:

- Character sequences of an arbitrary length
- Defined using single or double quotes
- "+" operator used to concatenate strings

Data types

Number:
- Unique data-types for floating-point and integer numbers
- Internally all represented as 64-bit floating point numbers

String:
- Character sequences of an arbitrary length
- Defined using single or double quotes
- "+" operator used to concatenate strings

Boolean:
- Values `true` or `false`
- Mapped to numbers 1 and 0

Functions

- Object defined using `function` keyword
- Properties and subfunctions only accessible from within the function itself
- Two types can be destinguished:

  Anonymous-functions:

  Nameless function, usually stored as property of an object or in a variable

  Named-functions:

  Function, that is given a specific name at definition time

Object handling

- Access to object-properties using:
  - Dot-notation: `object.property`
  - Array-notation: `object["property"]`
- No classes are provided
- New objects are created using the keyword `new` followed by a constructor function
- Predefined functions: *Object()*, *String(param)*, ...

Object handling

Example constructor function:

```
// constructor function BeerBottle
function BeerBottle(content){
    this["content"] = content;
    // fill property drink with function
    this.drink = function(){
        this.content = "empty";
    }
}
// creating new object; content set to Duff
var duffBeer = new BeerBottle("Duff");
// setting content to empty
duffBeer.drink();
```

Inheritance

Inharitance in JavaScript is realized using prototype-concept:

- Each object can have a prototype-object
- Treated as *super-object*
- Has to be set before object creation (constructor-function)
- Calls for properties that are not defined for an object will be forwarded to its prototype
- Recursive definition $\rightarrow$ possibility to define long prototype-chains

Inheritance

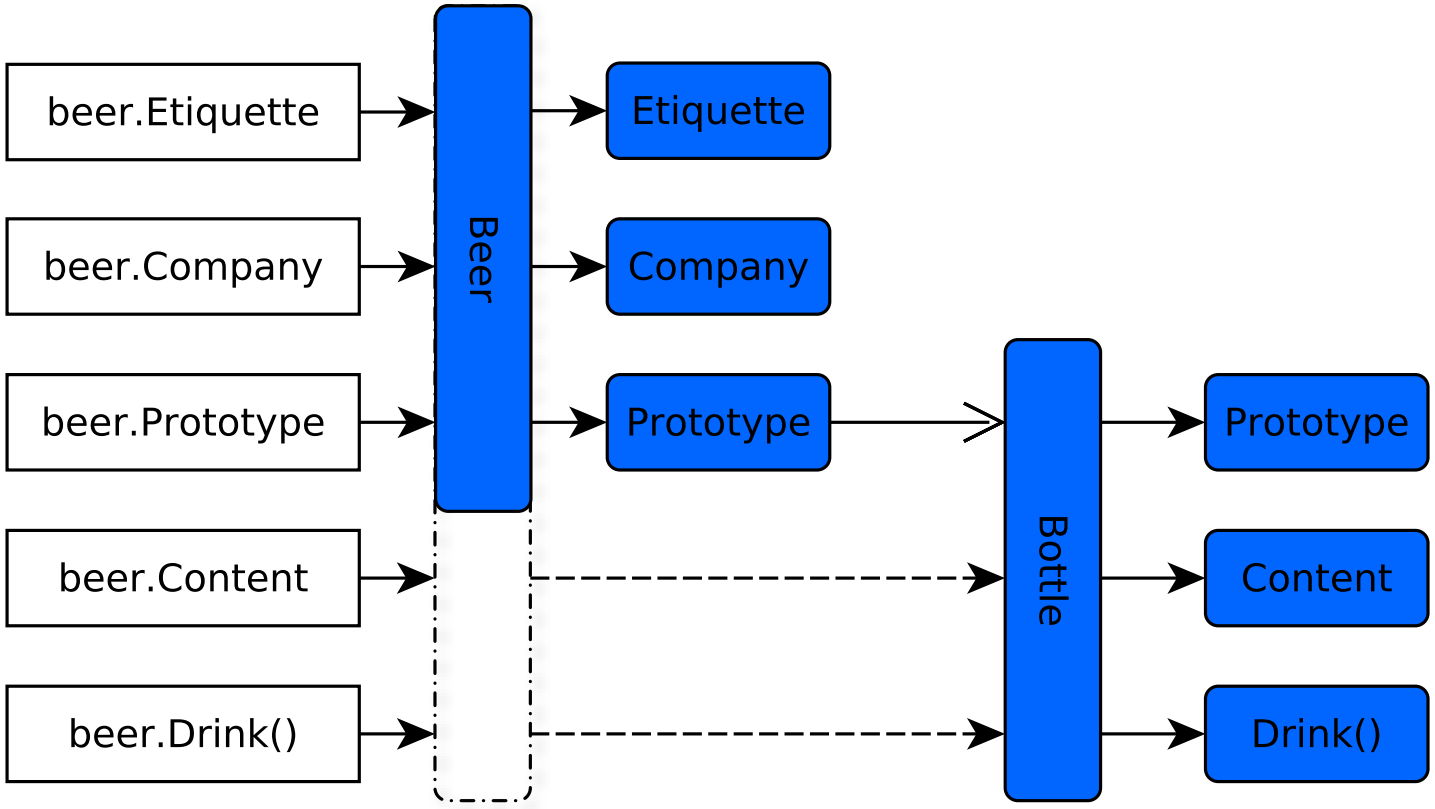

Figure : Concept of prototypes

Arrays

Arrays in JavaScript can be seen as normal objects, where all data represents a property of the array-object. These arrays have the following properties:

- Access using array-notation
- Length property always larger than the largest integer index
- Mixture of associative arrays and arrays known from C or Java
- Access using dot-notation to values that wheren't associated to numeric indexes
- Implemented as "sparse arrays"

```
var audience = new Object();
audience.askForQuestions = function(){
    document.write("Questions?");
}

audience.askForQuestions();
```