3

Development of Internet Applications

JavaScript

Ing. Michal Radecký, Ph.D.

What is JavaScript

Zdroj: http://petr.vaclavek.com

- Scripting programming language (interpreted) developed for dynamical behavior of WWW pages on the client side.
- Features
  - A part of HTML source code (DOM)
  - Multiplatform
  - Depended on interpreter (web browser, V8/NodeJS, atc…)
  - Objected oriented, class less (prototypes)
  - Case-sensitive
  - Similar syntax to C / C++ / Java / Python
  - Weakly typed
  - Nothing to do with Java

History of JavaScriptu

- Introduced in 1995 as part of Netspace Navigator (as a LiveScript).
- Microsoft in response to LiveScript introduced their own language called VBScriptu (only supported on Windows).
- In 1996 Microsoft introduced IE 3.0 with support of JScript (Microsoft implementation of ECMAScript Edition 3).
- In 1997 ECMAScript was standardized.
- ECMAScript is todays standard for JavaScript implementation (ESMAScript is standard and JavaScript is implementation of this standard).

What JavaScript can do

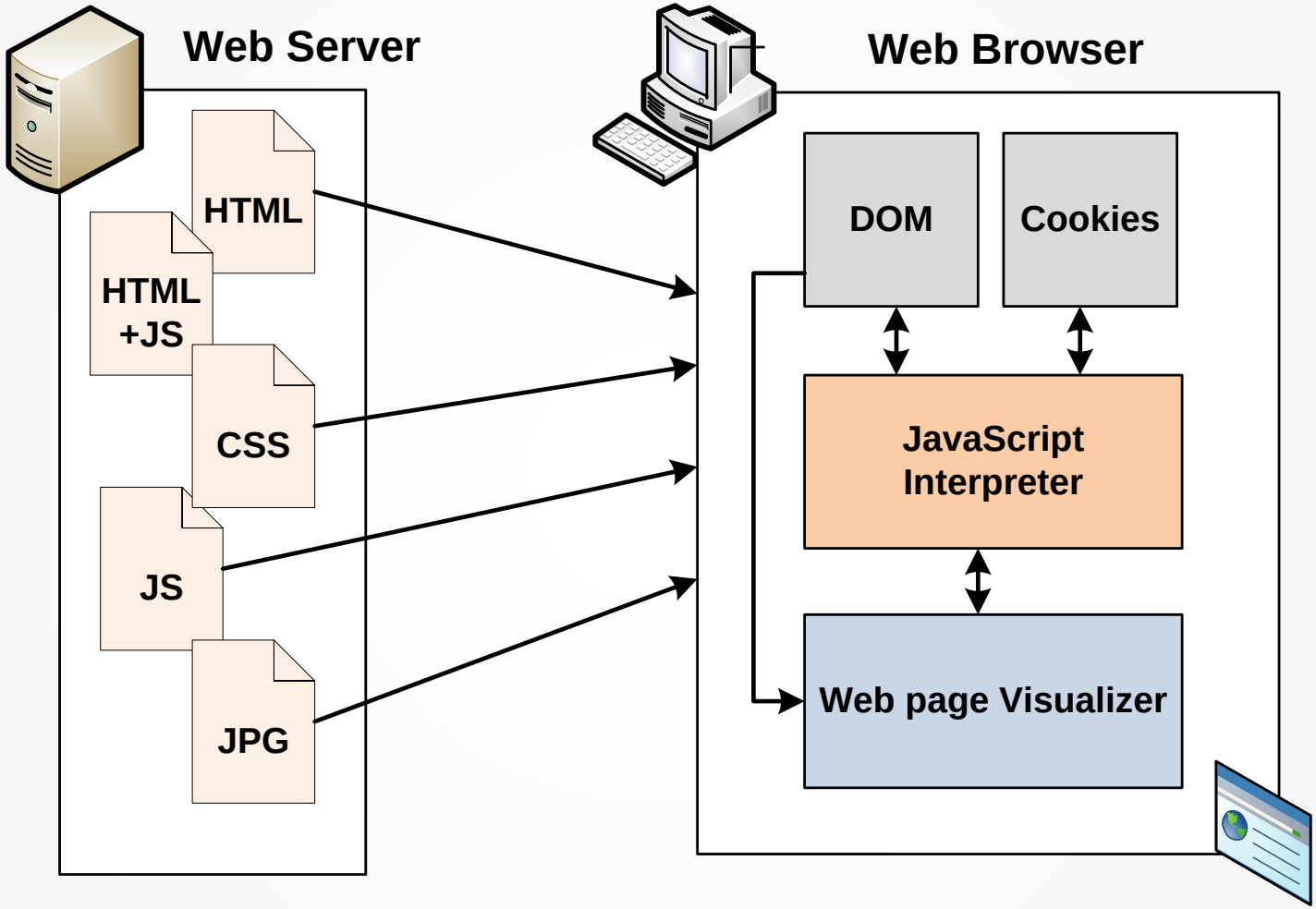
- Current JavaScript is powerful tool based on ideas of Perl, C/C++/Java or TCL.
- Influence of appearance and content of HTML document.
- Manipulation with images and other elements.
- Control web browser partially.
- Execute algorithms, calculations, etc.
- Control and manipulate with forms and theirs values.
- User events processing.
- Store and read a data in the form of Cookies.
- Collaborate with Flash, Java Applets and other plugins.

- Work environment of JavaScript is always limited by web browser (in the case, that we use JavaScript within the web browser).

What JavaScript can not do

- Draw vector graphics (it is not 100% true in HTML 5)
- Work directly with network resources.
- Read and write to the local files (it is not 100% true in HTML 5).
- Autonomously provide secure access to server (authentification and authorization) .
- Execute applications on OS level.
- Operate if user don't want it.

How JavaScript works

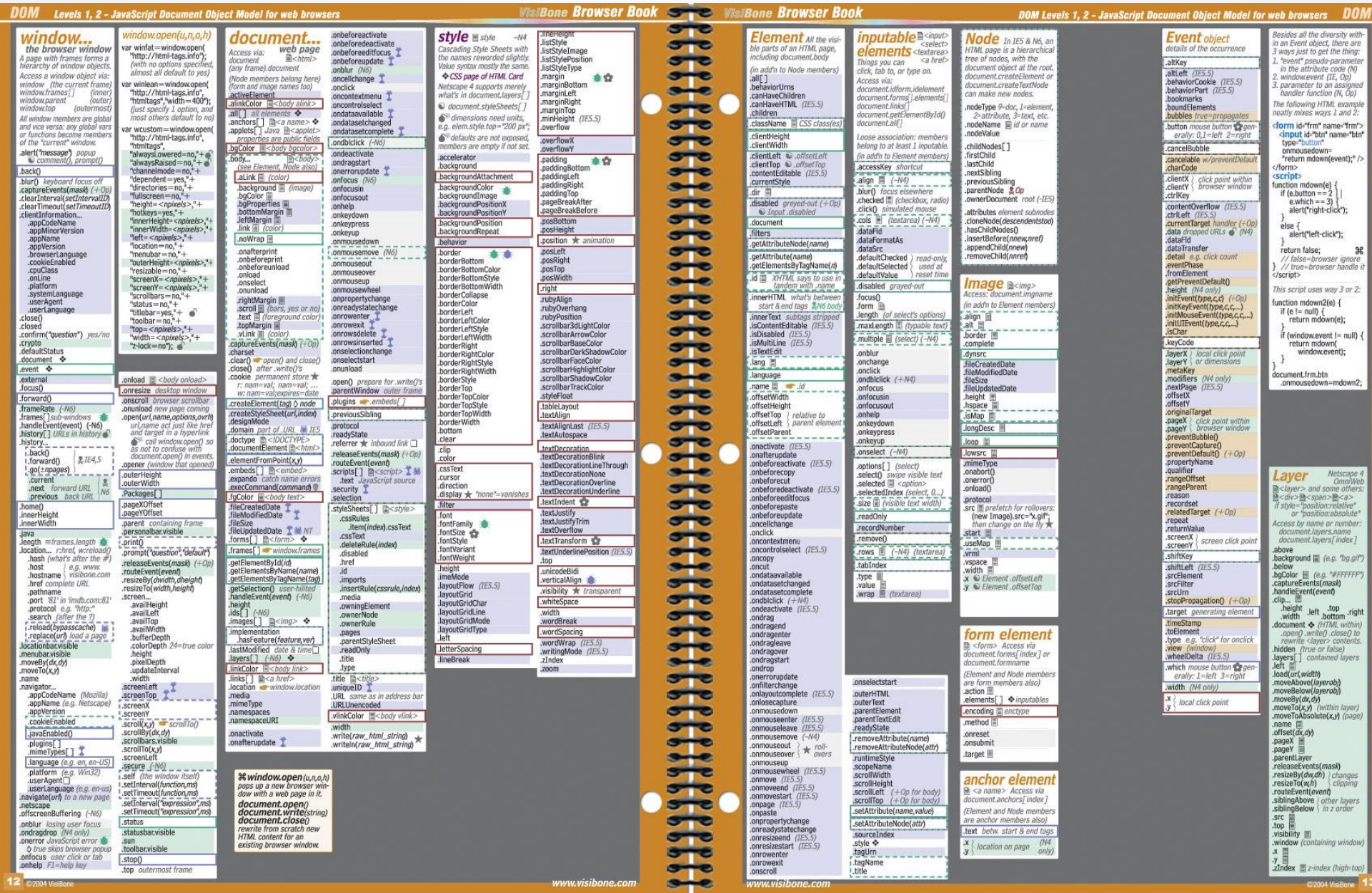


JavaScript is still in the form of source code (similar to HTML source code).

DOM (Document Object Model)

- Object-oriented representation of an XML or HTML document.
- It is an API for object-oriented access to individual elements of a web page and their attributes, methods, etc.
- The tree data structure is used.
- W3C DOM standard, formerly Intermediate DOM (Netscape, document.layers) and DHTML OM (Microsoft, document.all).
- The standard distinguishes Levels (0-3), which specify a set of features and functions that the DOM of a given level must satisfy.

JavaScript DOM



JavaScript in page

Zdroj: http://petr.vaclavek.com

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
    <title>My first JavaScript powered page</title>
      <script type="text/javascript" src="library.js"></script>
    <script type="text/javascript">
            alert („Hello world!");
    </script>
      <noscript>
              This part is displayed if JavaScript is disabled.
      </noscript>
  </head>

  <body onload="alert(‚Loaded!')">
     Standard HTML content
       <a href="javascript: alert (one plus one is: '+(1+1));">1+1=?</a>
  </body>
</html>
```

onLoad – is dispatched when page and all resources are loaded (images, styles, atc…)

JavaScript constructions

```
document.write(“Hello“);
document.write(„These 'are' quotas");
document.write(„These \“are\" quotas“ + “ – again”);
Console.log(a);
```

```
var p1 = 10;
var p2 = “10.5”;
p3 = “hello”;
var p4 = true;
document.write(p1 + p2); //1010.5
p2 = 10.5;
document.write(p1 + p2); //20.5
```

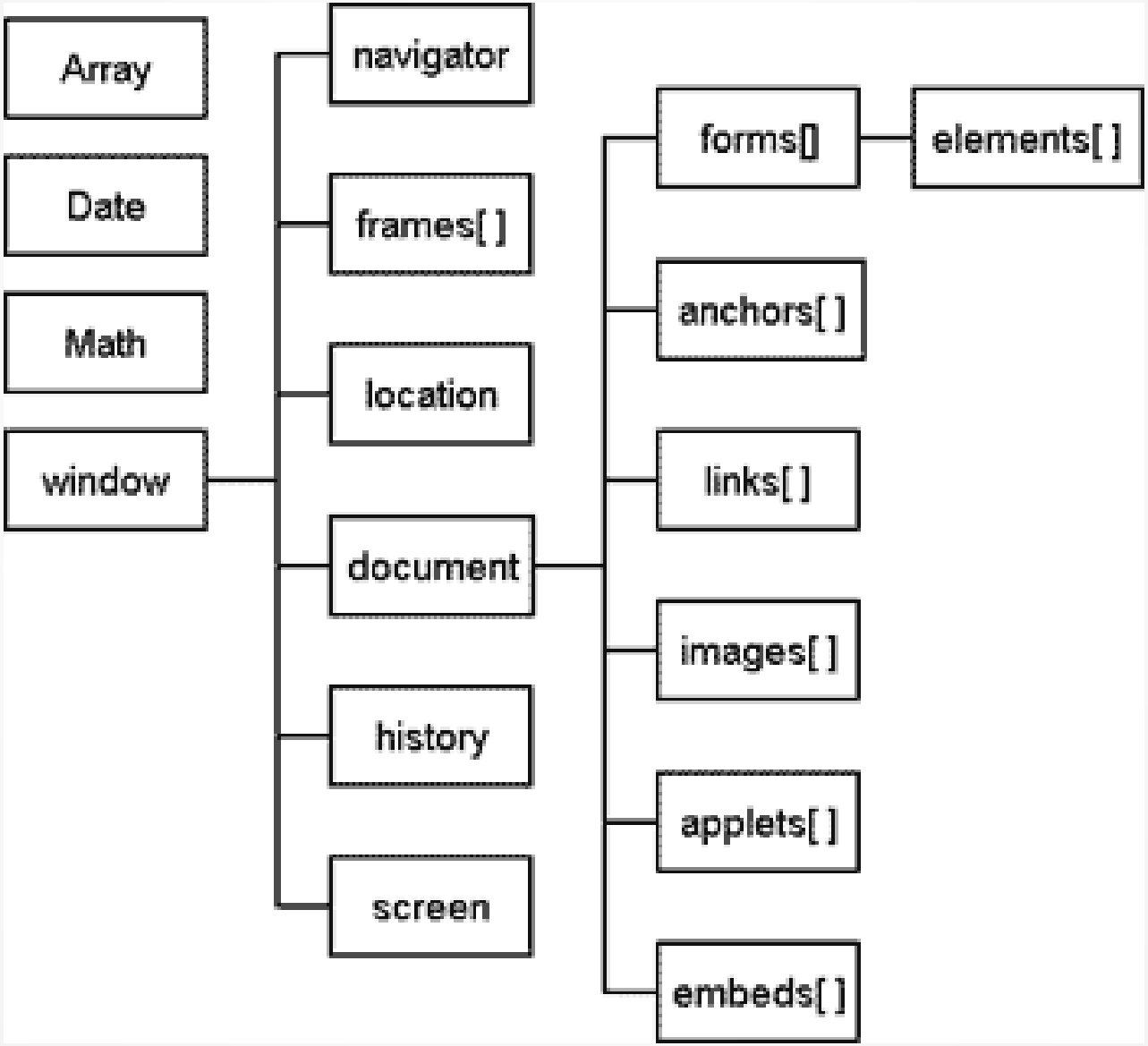
```
var array2 = ["carrot", "potatoes", "cauliflower"] //std. one-dimensional
for(i=0; i < array2.length; i++){
   document.write(array2[i] +" ")
}

array2[“br”] = “potatoes”;

var array = new Array("HTML", "DHTML", "XHTML");
document.write(array.valueOf()); //HTML,XHTML,XHTML
document.write(array.toSource()); //"["HTML", "DHTML", "XHTML"]"
```

Basic objects

Zdroj: http://petr.vaclavek.com

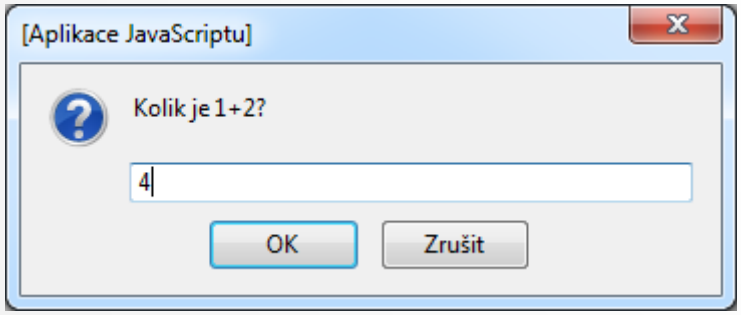


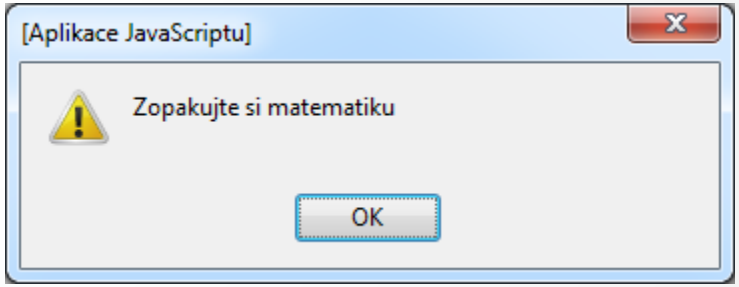
window.

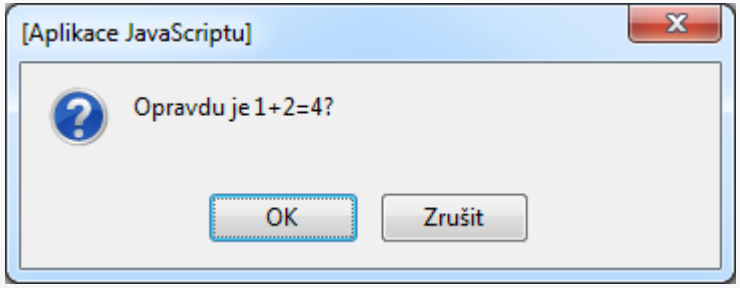
Zdroj: http://petr.vaclavek.com

```
    var result = prompt („How much is 1+2?", "4");
if (result){
  var conf = confirm („Confirm that 1+2=" + result + "?");

  if (conf){
    alert („Go to school");
  } else
    alert („Right!");
}
```







location. and history.

Zdroj: http://petr.vaclavek.com

```
<script type="text/javascript">
<!--
function delayer(){
    window.location = "http://www.cs.vsb.cz";
}
//-->
</script>
...
<body onLoad=„window.setTimeout('delayer()', 5000)">
...
```

```
<a href="javascript:history.back();">Zpět</a>
<a href="javascript:history.forward();">Vpřed</a>
<script type="text/javascript">
      <!–
    if (document.referrer != '')
        document.write (,Your refferer is <a
   href="'+document.referrer+'">'+document.referrer+'</a>');
      else
        document.write ('History does not contain any items or you work with local web page.');
      // -->
</script>
```

navigator.

Zdroj: http://www.javascriptkit.com

```
<script type="text/javascript">

if (/MSIE (\d+\.\d+);/.test(navigator.userAgent)){ //test for MSIE x.x;
 var ieversion=new Number(RegExp.$1) // capture x.x portion and store as a number
 if (ieversion>=8)
  document.write("You're using IE8 or above")
 else if (ieversion>=7)
  document.write("You're using IE7.x")
 else if (ieversion>=6)
  document.write("You're using IE6.x")
 else if (ieversion>=5)
  document.write("You're using IE5.x")
}
else
 document.write("n/a")
</script>
```

Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0)

Document.

- Individual element events (onclick, onmouseover, onload, onsubmit, etc.)
- DOM querying (recursively)
  - getElementById
  - getElementsByTagName
  - getElementsByClassName
  - querySelector, querySelectorAll
- DOM creation and modification
  - innerText, innerHTML
  - createElement, createTextNode
  - appendChild

Objects

Zdroj: http://www.augi.cz/programovani/javascript-ocima-programatora/

```
var car = {   // anonymous object
    name : "Honda",
    model : "Civic",
    owner : { name : "Jiri", surname : "Novak" },
    printMe : function() {
        return this.name + ' ' + this.model + ' owned by ' + this.owner.name + ' ' + this.owner.surname;
    },
};
```

```
function Car(carName, model) { // constructor
    this.name = carName;
    this.model = model;
    this.printMe = function() {
        return this.name + ' ' + this.model;
    };
}

var car1 = new Car("skoda", "fabia");
```

Inheritence is not directly supported, there are prototype or variables relations.

Objects

Zdroj: http://www.augi.cz/programovani/javascript-ocima-programatora/

```
var hc = new Car();
var sf = new Car("Skoda", "Fabia");

// ensure that all objects created with Car have attribute spz
Car.prototype.spz = ,first';
document.write(hc.spz); // ,first'
document.write(sf.spz); // ,first'

// the prototype is not taken into account during assignment
hc.spz = ,second';
document.write(Car.prototype.spz); // 'second'
document.write(hc.spz); // ,second'
document.write(sf.spz); // ,second'

// of course, if we assign to the prototype...;)
Car.prototype.spz = ,third';
document.write(Car.prototype.spz); // ,third'
document.write(hc.spz); // ,second'
document.write(sf.spz); // ,third'
```

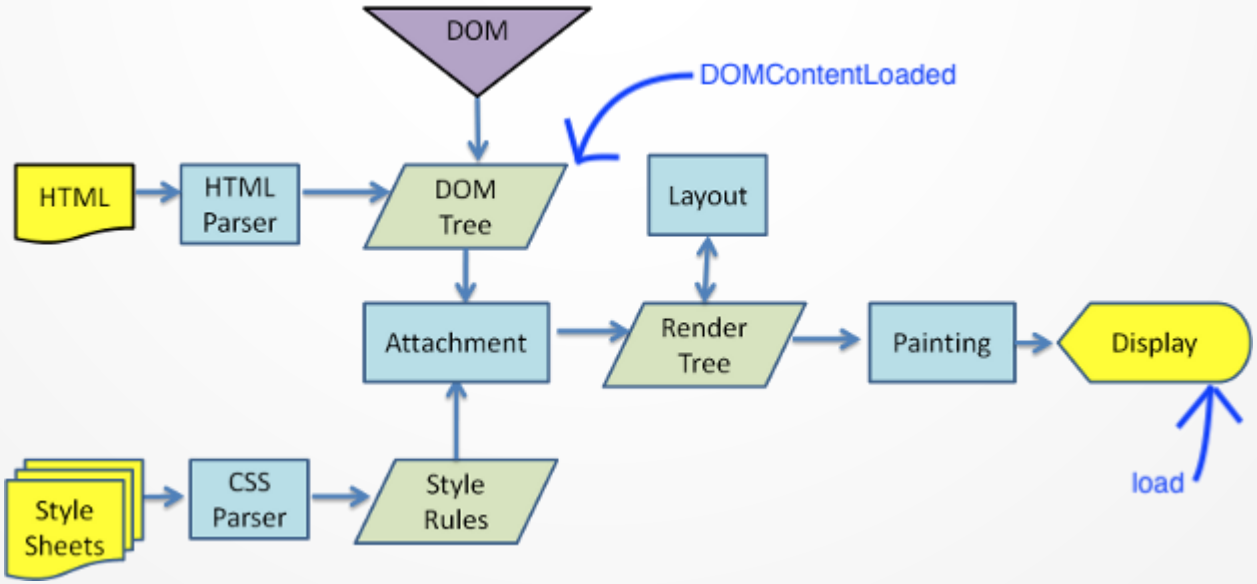
Prototyp is a part of each object that is the same for all objects created using the same descrip
First, the particular object is checked, than the prototype.

DOM events

```
window.addEventListener("load", (event) => {
  console.log("page is fully loaded");
});

document.addEventListener("readystatechange", (event) => {
  console.log("page DOM is ready in different states");
});

document.addEventListener("DOMContentLoaded", (event) => {
  console.log("page DOM is loaded with defered scripts etc.");
});
```



Zdroj: https://javascript.info/callbacks/

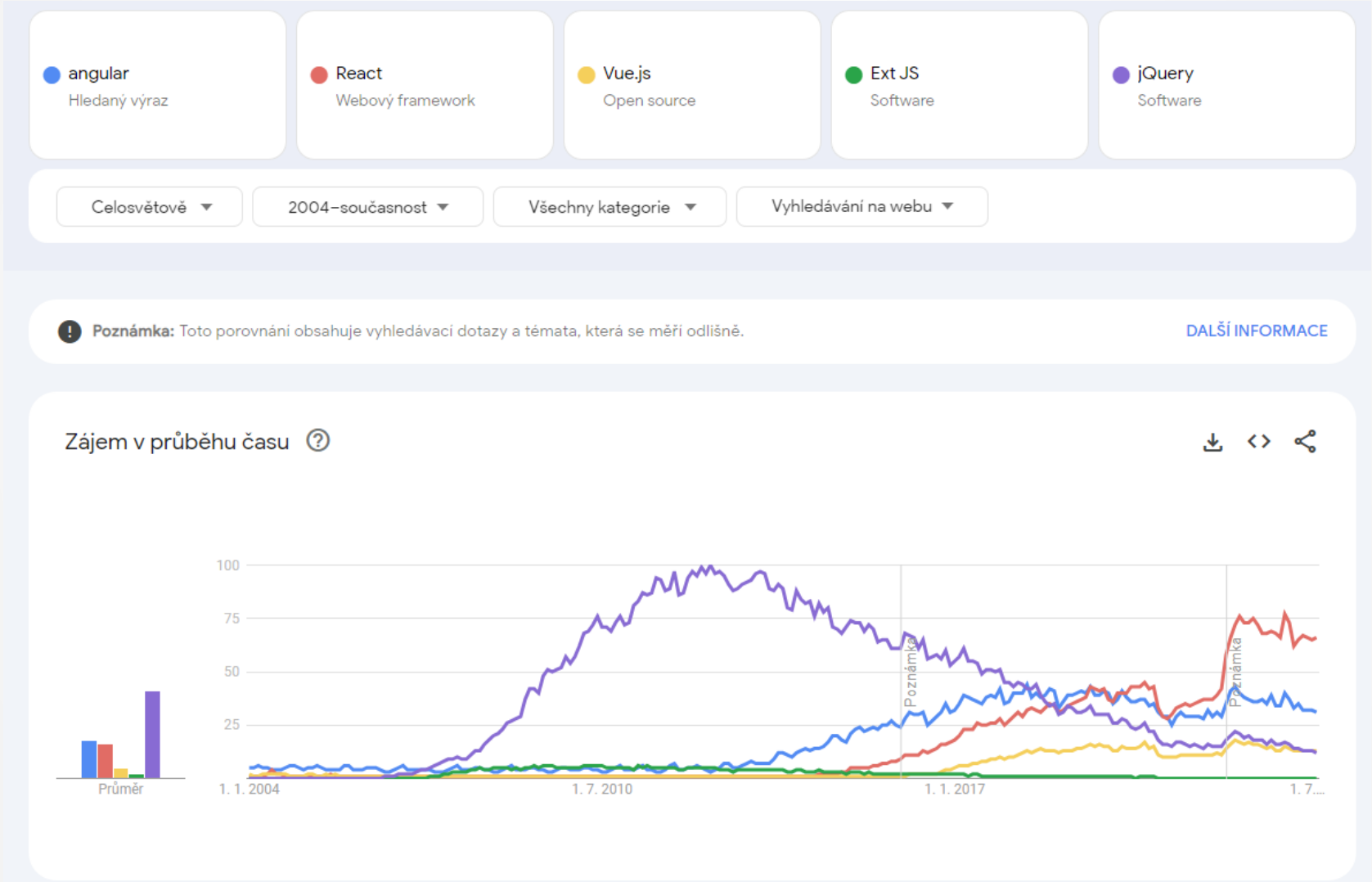
Asynchronous programming

- Events – EventListener
  - no control over order of processing
- Callback functions
  - chaining of callback funkcí (callback hell), immediately firing of methods, only one invocation
- Promise objects, async/await
  - Functional approach
  - Future promise of value, independent of time, better error handling, multiple invocation
  - async/await – automatization of Promise constructions

JavaScript frameworks

- They are JavaScript libraries which help with development of applications and make the work easier.
- The developer can be more focused on solving of problems, not on the optimization and debugging of the code for all web browsers.
- They are based on pure JavaScript and extend the objects, methods, etc. (by usage of prototype)
- There are two basic groups
  - JavaScript libraries – functionality extensions (Prototype, jQuery, MooTools, script.aculo.us,)
  - RIA frameworks – complex solutions for RIA based on JS (Angular.js, Backbone, React.js, Embed, YUI, Dojo, extJS, GWT)

JavaScript frameworks



jQuery

```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <script src="http://code.jquery.com/jquery-2.1.4.min.js" type="text/javascript"></script>
  <script type="text/javascript">
    $(document).ready(function(){
      $("a").click(function(event){
        alert("As you can see, the link no longer took you to jquery.com");
        event.preventDefault();
      });
    });

  </script>
</head>
<body>
  <a href="http://jquery.com/">jQuery</a>
</body>
```

```
$(document).ready(function() {
  $("#orderedlist li:last").hover(function() {
    $(this).addClass("green");
  },function(){
    $(this).removeClass("green");
  });
});
```

$(document).ready – dispatch when DOM is ready (do not wait for resources like images, styles, etc..)