VEEBIRAKENDUSTE LOOMINE

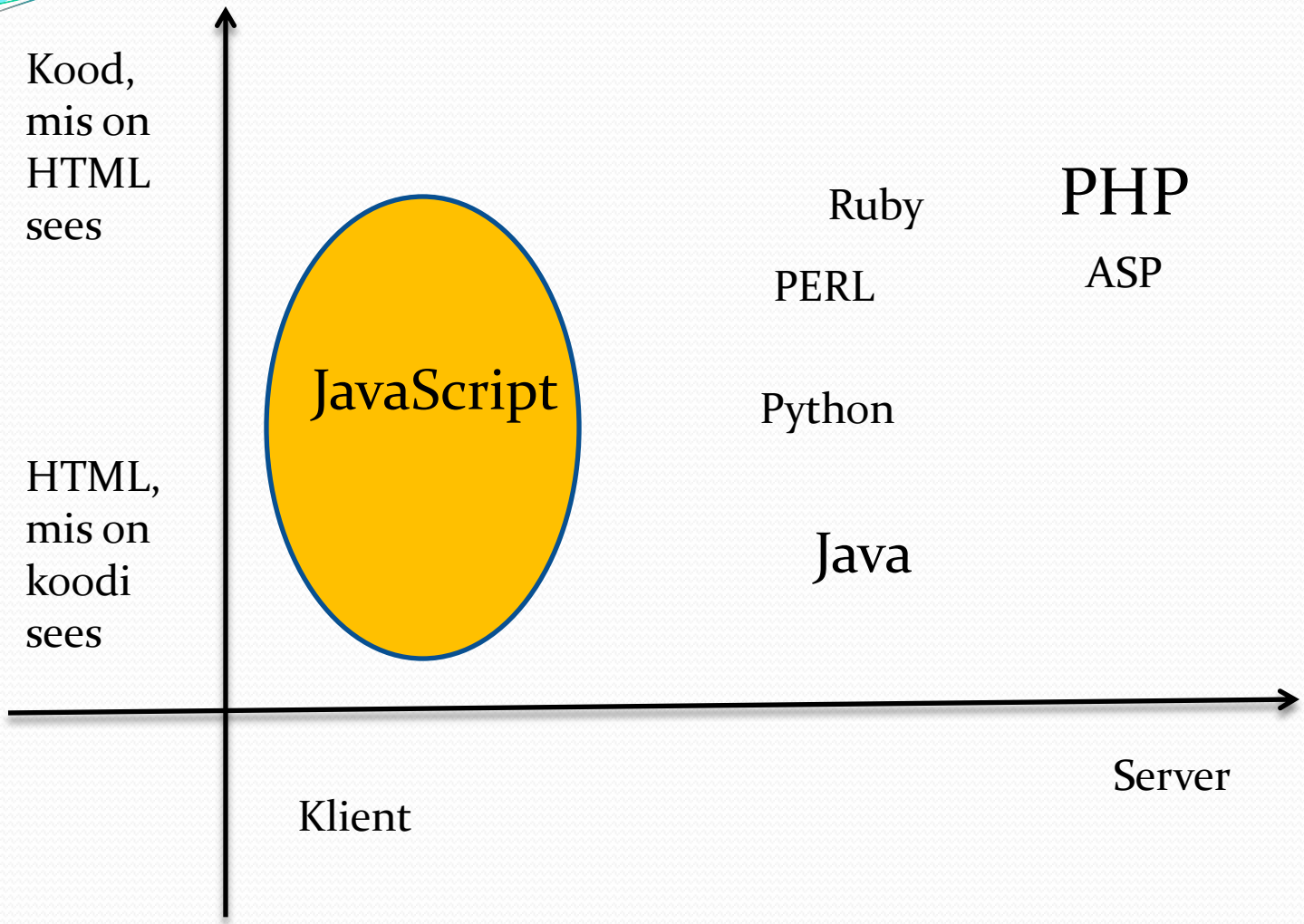
MTAT.03.230 (6EAP)

3. Loeng

Helle Hein

JavaScript ja DOM

Kus me oleme?


Kood, mis on HTML sees
HTML, mis on koodi sees
JavaScript
Ruby
PERL
Python
Java
PHP
ASP
Server
Klient

Ajalugu

Sun: Java

- Netscape: LiveScript (Brendan Eich) /JavaScript

(Navigator 2.0 - Detsember 1995)

- Microsoft: JScript (IE 3.0 August 1996)

- ECMA (European Computer Manufacturers Association)

Standard: ECMAScript

(ECMA 262 ver 3 – 1999, ver 5 – 2009, …)

- JavaScript “serious” language since

IE6, Firefox1 (stable “enough” platform)

Versioonid

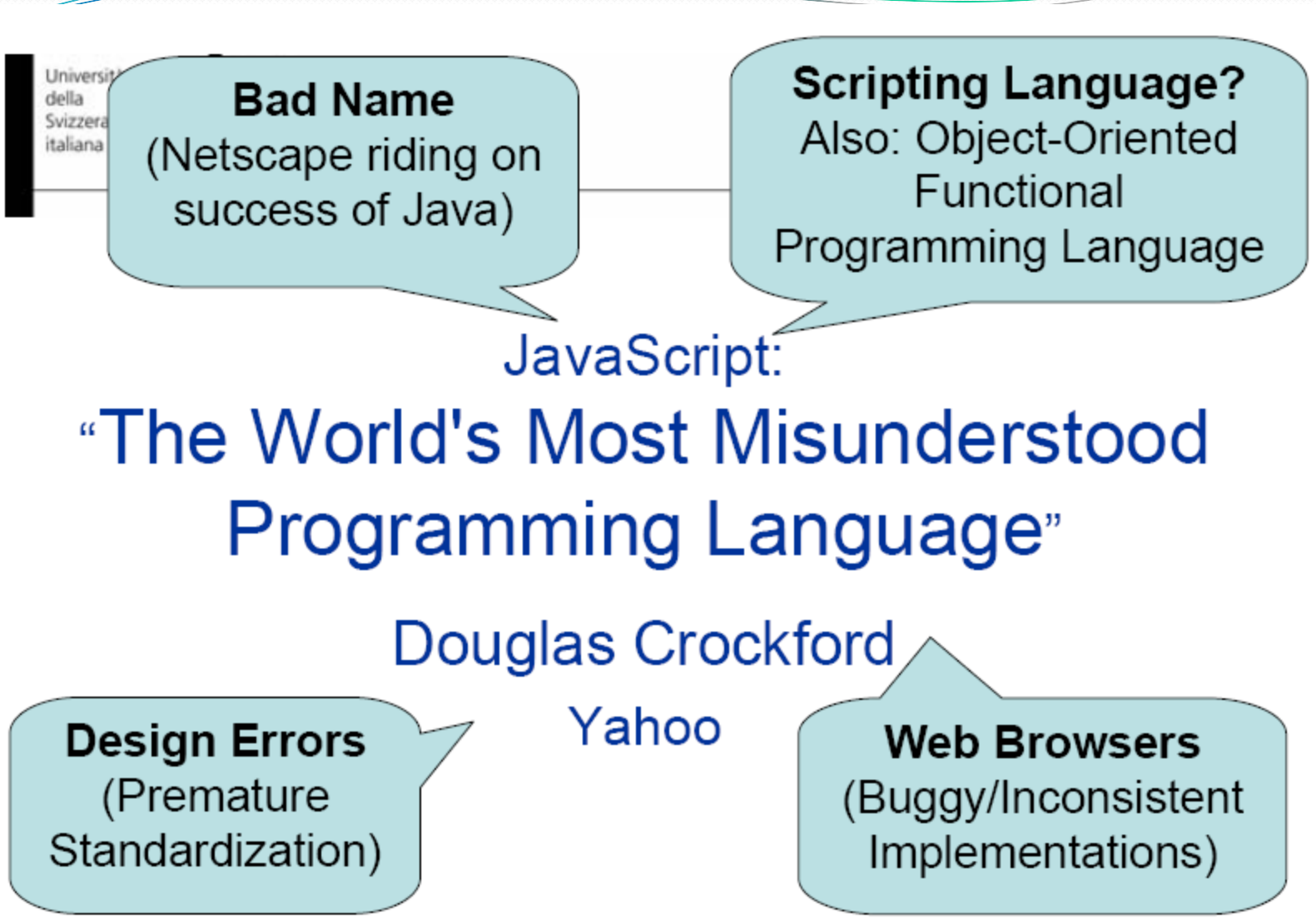
Browser	Year	JavaScript Version
Netscape Navigator 2.0	1995	JavaScript 1.0
Microsoft Internet Explorer 3.0	1996	JavaScript 1.0 (JScript 1.0)
Netscape Navigator 3.0	1996	JavaScript 1.1
Netscape Navigator 4.0	1997	JavaScript 1.2
Microsoft Internet Explorer 4.0	1997	JavaScript 1.2 (JScript 3.0)
Netscape Navigator 4.5	1998	JavaScript 1.3
Microsoft Internet Explorer 5.0	1999	JavaScript 1.3 (JScript 5.0)

Versioonid

Browser	Year	JavaScript Version
Netscape Navigator 6.0	2000	JavaScript 1.5
Microsoft Internet Explorer 6.0	2000	JavaScript 1.3 (JScript 5.0)
Microsoft Internet Explorer 7.0	2006	JavaScript 1.3 (JScript 5.0)
Netscape Navigator 7.0	2007	JavaScript 1.6
Microsoft Internet Explorer 8.0	2009	JavaScript 1.5(JScript 5.0)

Versioonid

- Firefox 1.0 toetab JavaScript 1.5 (ECMAScript 3 ekvivalent)
- Firefox 1.5 toetab JavaScript 1.6 (1.5 + Array Extras + E4X +…)
- Firefox 2.0 toetab JavaScript 1.7 (1.6 + Generator + Iterators + …)
- Firefox 3.0 toetab JavaScript 1.8 (1.7 + Generator Expressions +…)


Università della Svizzera italiana
Bad Name
(Netscape riding on success of Java)
Scripting Language?
Also: Object-Oriented
Functional
Programming Language
JavaScript:
"The World's Most Misunderstood Programming Language"
Douglas Crockford
Yahoo
Design Errors
(Premature Standardization)
Web Browsers
(Buggy/Inconsistent Implementations)

Mida JavaScript võimaldab

- Veebilehe reageerimist kasutaja tegevusele kasutades form elemente (input fields, text areas, buttons, radio buttons, checkboxes, selection lists) ja hypertext linke

- Anda kasutaja käsutusse väikeses koguses infot ja sõbralikku kasutajaliidest

- Juhtida navigeerimist, plug-in’e või Java rakendeid lähtudes kliendi valikust

- Töödelda kliendi andmeid enne serverisse saatmist

- Muuta dünaamiliselt sisu ja stiili

Mida JavaScript ei võimalda

- Muuta brauseri seadeid
- Käivitada rakendust kliendi arvutis
- Lugeda või kirjutada faile (katalooge) kliendi või serveri masinas
- Saada kätte andmevoogu serverist
- Saata salajasi e-meile veebisaidi külastajatelt
- Graafikat (va HTML dünaamiline genereerimine)

Skriptimiskeel

- Lisada staatilisele lehele dünaamilist käitumist
  - `(X)HTML` + `CSS` on deklaratiivsed (elementide välimus, paigutus, ..)
  - Veebilehe genereerimine dünaamiliselt
  - Veebilehe sisu muutmine

- Käivitada skript brauseris
  - Minimeerida reageerimisaega kasutajaga suhtlemisel
  - Vahetu tagasiside (*input form validation)*
  - Vahetu reageering kasutaja sündmustele
  - Mõjutada brauseri tööd (history, window, popups, statusbar)

JavaScript põhiomadused

- JavaScripti lähtekood laetakse alla ja käivitatakse brauseri poolt
• Tüüpimata
– Dünaamilised andmetüübid
• Objektid konteineritena
– **Puuduvad klassid**
– Objektid on sarnased “HashMap” -idele
• “Prototype” pärimine
– Erineb klassikalisest objekt-orienteeritud pärimisest
• Lambda
– Funktsioonid on ka objektid (nagu funktsionaalsetes keeltes)

Kuhu JavaScript kood panna?

- Kaks võimalust:
  - Otse HTML faili

```
<script type="type/javascript">
        //kood siia
</script>
```

  - Eraldi faili

```
<script src="URL"
        type="type/javascript">
</script>
```

    Faili laiend **.js**
    - Sisaldab ainult JavaScript lauseid

JavaScript vs. Java

JavaScript vs Java

JavaScript

–Tüüpimata

– Objektid

– Üks globaalne skoop ainult

– Lõimed puuduvad

– Objekti-põhised erindid

• JavaScript Runtime

– Kood juhib kogu veebilehte
– Lähtekoodi interpreteerib brauseri VM

Java

– Rangelt tüübitud

– Klassid & Objektid

– Paketid

– Mitmelõimelisus

– Kontrollitud erindid

• Java Runtime

– Rakend töötab kasti sees(või uues aknas)
– kompileeritakse baitkoodi, mida interpreteerib JVM

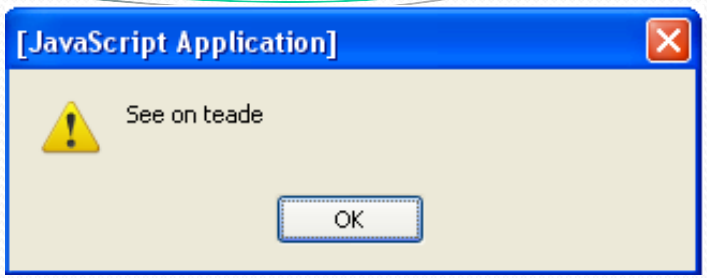
Iga JavaScript implementatsioon sisaldab kahte tarkvara komponenti:

- skriptimootor (scripting engine),
- hosti keskkond (hosting environment).

native objekt - objekt, mis on nõutud ECMAScripti definitsiooni poolt ja seetõttu pakuvad skriptimootorid (nt. `Object, Math, Number, Array, Boolean, …`);

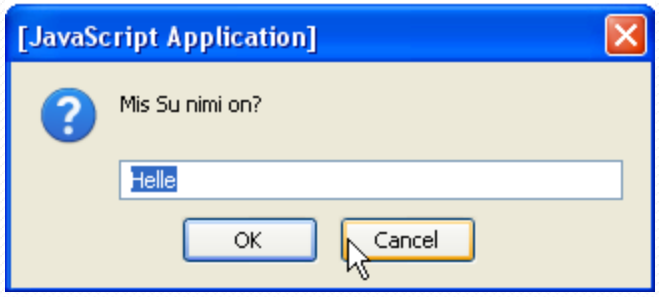
hosti objekt - objekt, mida pakuvad keskkonnad (`alert, prompt, document, …`).

```
window.alert("See on teade");
```




```
window.confirm("Kas oled kindel, et andmed on õiged?");
```


```
window.prompt("Mis Su nimi on?", "Helle");
```



Kuidas JavaScript töötab – sammud

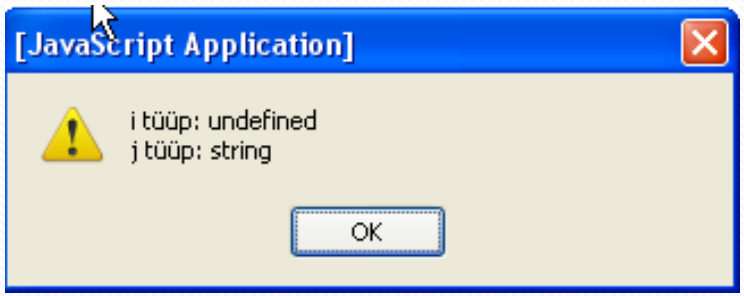
1. Brauser laeb (loads) veebilehe
2. Brauser leiab JavaScripti
3. Brauser edastab koodi interpretaatorile
4. Interpretaator täidab koodi
5. Kood võib veebilehega manipuleerida
6. Brauser näitab veebilehte

JavaScript andmetüübid

Igal muutujal on väärtus ja iga väärtus kuulub ühte kuuest JavaScript andmetüübist:

- Number
- String
- Boolean
- Object
- **Null**
- **Undefined**

```
var i;
var j;
j = "ei ole arv";
alert("i tüüp: " + (typeof i) + "\n" +
      "j tüüp: " + (typeof j));
```



JavaScript andmetüübid

Teisendusreeglid näiteks `Number` jaoks:

Originaalväärtus	**`Number` väärtus**
Defineerimata	`NaN`
null, false, „“	0
true	1
NaN	NaN
Arv stringina	esitatud arv
Mingi sõne	NaN
Objekt	Kutsutakse välja `valueOf()` meetod

JavaScript objektid

❖ Objekt on omaduste (properties) hulk, mis koosneb unikaalsest nimest koos väärtusega (kuus JavaScript tüüpi).

```
var o = new Object();
o.prop = true;
o.prop = “true“;
o.prop = 1;
```

❖ JavaScript programmid ei defineeri klasse.

```
var o1 = new Object();
o1.testing = “See on test“;
delete o1.testing;
```

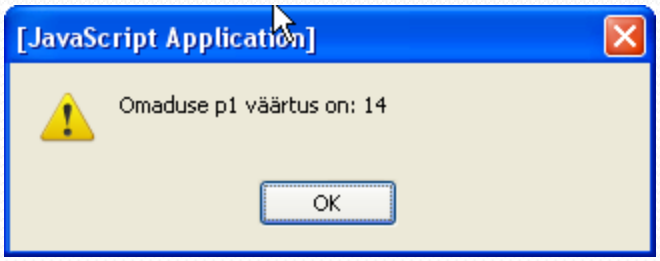
❖ Kui omadust pole veel loodud, siis ta luuakse.

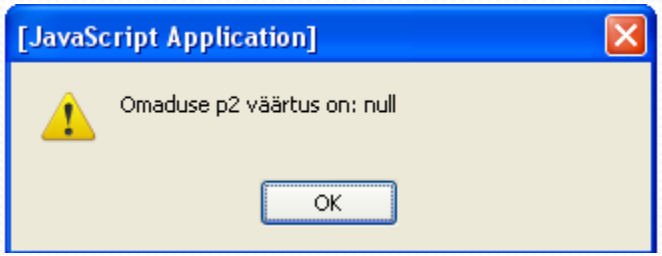
```
var o2 = {p1:5+9, p2:null,testimine:"See on test"};
```

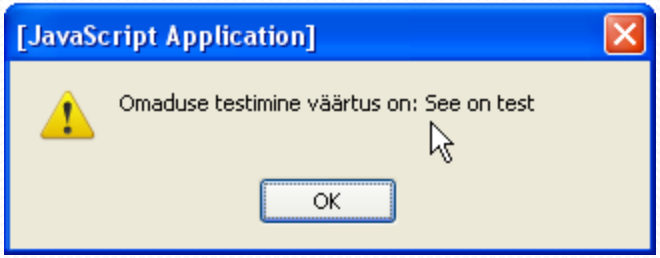
JavaScript objektid

Omaduste läbivaatamine

```
var o2 = {p1:5+9, p2:null,testimine:"See on test"};
for (var aName in o2){
  window.alert("Omaduse " + aName + " väärtus on: " + o2[aName])
}
```







Objekt ja massiiv

Objekti omaduste väärtustele saab ligi kahel viisil:

`o2.p1` **või** `o2["p1"]`

JavaScript objekti võib vaadelda kui *assotsiatiivset* massiivi (sarnane klassile `java.util.Hashtable`)

```
var a1 = new Array();
```

Massiivi loomine konstruktori abil:

```
var a2 = new Array(4, false, "OK");
```

JavaScript massiiv on sarnane Java klassile `java.util.Vector`

Massiivi meetodeid

Meetod	Kirjeldus
`toString()`	Tagastab Stringina massiivielemendid komaga eraldatult.
`sort(Object)`	`Object` argument on funktsioon, mis täpsustab sortimisjärjekorra.
splice(Number,0,any type)	Massiivi lisatakse kolmas argument esimese argumendiga näidatud indeksi kohale, ülejäänud nihutatakse.
`splice(Number, Number)`	Massiivist eemaldatakse teise argumendiga näidatud arv elemente alates esimese argumendiga näidatud indeksist.
`push(any type)`	Lisatakse lõppu antud element. Tagastatakse massiivi pikkus.
`pop()`	Eemaldatakse viimane element. Tagastatakse element.
`shift()`	Eemaldatakse esimene element. Ülejäänud nihutatakse. Tagastatakse element.

Objektid JavaScriptis

❖ **Objektid, mis defineeritakse seoses HTML märgistega.**
link, tekstiväli, nupp, …
JavaScripti abil on kättesaadav iga objekt HTML dokumendis.

❖ **Objektid, mis defineeritakse automaatselt brauseri poolt.**
`navigator`, `history`, ..

❖ **Objektid, mis on JavaScripti sisse ehitatud, nt** `Date` **ja** `Number`**.**

❖ **Omaloodud objektid kasutades operaatorit** `new`**.**

`http://www.javascriptkit.com/jsref/window.shtml`

JavaScript funktsioonid

- JavaScriptis esitatakse iga funktsioon objektina.

- Funktsiooni deklaratsiooni töödeldes luuakse objekt ja funktsiooni nimega muutuja, mis hakkab viitama sellele objektile.

- Meetod on JavaScriptis funktsioon, mis on omistatud objektile omaduse väärtusena.

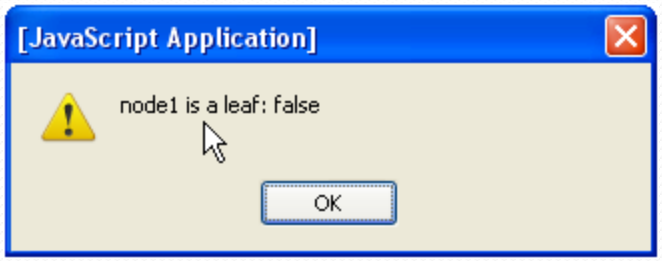
```
function name(parameters) {
…
return [expression];
}
```

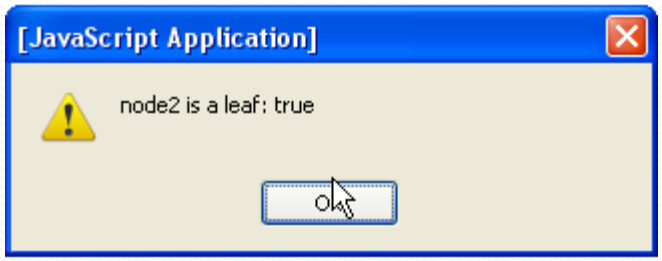
JavaScript funktsioonid

```
function leaf() {
  return this.left == null && this.right == null;
}
function makeBTNode(value) {
  var node = new Object();
  node.left = node.right = null;
  node.value = value;
  node.isLeaf = leaf;
  return node;
}
var node1 = makeBTNode(3);
var node2 = makeBTNode(7);
node1.right = node2;
window.alert("node1 is a leaf: " + node1.isLeaf());
window.alert("node2 is a leaf: " + node2.isLeaf());
```

JavaScript funktsioonid

```
function leaf() {
  return this.left == null && this.right == null;
}
function makeBTNode(value) {
  var node = new Object();
  node.left = node.right = null;
  node.value = value;
  node.isLeaf = leaf;
  return node;
}
var node1 = makeBTNode(3);
var node2 = makeBTNode(7);
node1.right = node2;
window.alert("node1 is a leaf: " + node1.isLeaf());
window.alert("node2 is a leaf: " + node2.isLeaf());
```





JavaScript funktsioonid

Et vältida segadust ja meetodi otsekasutust funktsioonina, on võimalik meetod defineerida omaduse sisse:

```
node.isLeaf =
function leaf() {
  return this.left == null && this.right == null;
};
```

Nüüd ei ole enam võimalik pöörduda otse `isLeaf()` poole.

JavaScript funktsioonid

- Igat funktsiooni võib välja kutsuda nii funktsioonina, meetodina kui ka konstruktorina.
- Iga JavaScripti funktsioon on ka konstruktor.

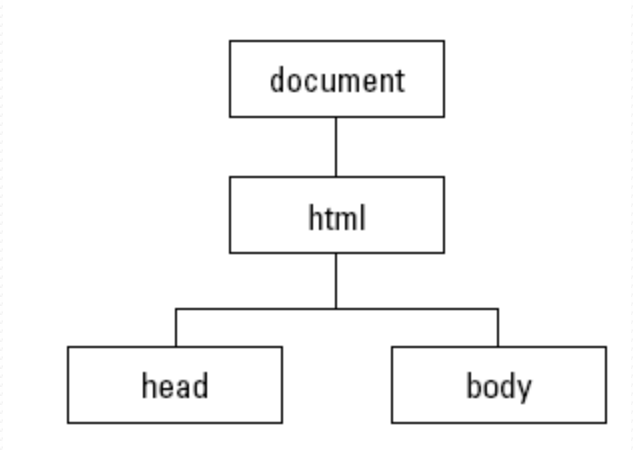
```
function BTNode(value) {
  this.left = this.right = null;
  this.value = value;
  this.isLeaf =
    function leaf() {
      return this.left == null && this.right == null;
    };
}
var node1 = new BTNode(3);
var node2 = new BTNode(7);
node1.right = node2;
window.alert("node1 is a leaf: " + node1.isLeaf());
window.alert("node2 is a leaf: " + node2.isLeaf());
```

JavaScript funktsioonid

```
function square(x) { return x*x; }
var a = square(4); // 16

var b = square; // b viitab samale funktsioonile
var c = b(5); // c =25

var o = new Object;
o.square = new Function("x", "return x*x");
y = o.square(16); // y = 256

var a = new Array(3);
a[0] = function(x) { return x*x; }
a[1] = 20;
a[2] = a[0](a[1]); // a[2] = 400
```

DOM - Document Object Model

Defineerib, kuidas JavaScript programmid saavad brauseri poolt näidatava HTML lehega manipuleerida.

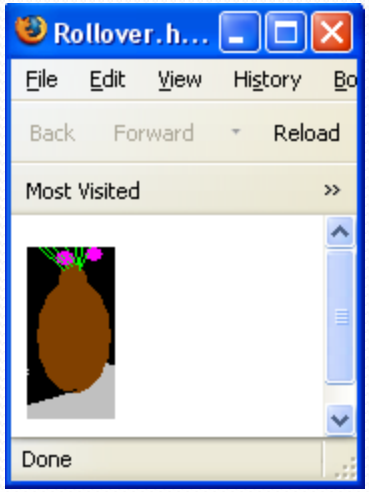


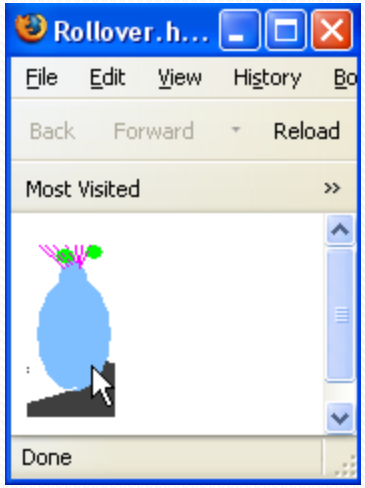

```
<html>
   <head></head>
   <body></body>
</html>
```

Vaatleme HTML dokumenti, kus hiire üleminekul pildist peab pilt muutuma:

```
<!DOCTYPE html
      PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
    <title>Rollover</title>
    <script type="text/javascript" src="rollover.js">
    </script>
    <meta http-equiv="Content-Script-Type"
       content="text/javascript" />
  </head>
  <body>
    <p>
      <img id="img1" src= "pic2.png" alt="flower pot"
        height="86" width="44"
        onmouseover="swap('img1', 'pic1.png');"
        onmouseout="swap('img1', 'pic2.png');" />
    </p>
  </body>
</html>
```

```
// rollover.js
function swap(eltId, URL) {
  var elt = window.document.getElementById(eltId);
  elt.src = URL;
//elt.setAttribute(„src“, URL);
  return;
}
```





DOM ajalugu ja tasemed

• Netscape 2.0 - meetod write() teksti lisamiseks dokumenti; samuti juurdepääs ankurelemendile ja vormile.

• Netscape 4.0 ja IE brauserid avasid kogu dokumendi skriptidele, kuid need brauserid tõlgendasid dokumendi mudelit erineval viisil.

• W3C DOM soovitus oktoobris 1998 – tase 1 (moodulid *core* ja *HTML*)

• W3C DOM soovitus november 2000 – tase 2 (lisaks sündmused ja stiilid)

• W3C DOM soovitus aprill 2004 – tase 3 (*Views ; Traversal and Range*)

Objekt `Window`

- On igal brauseri aknal ja raamil – globaalne objekt
- Sisaldab omadusi ja meetodeid

Omadused:

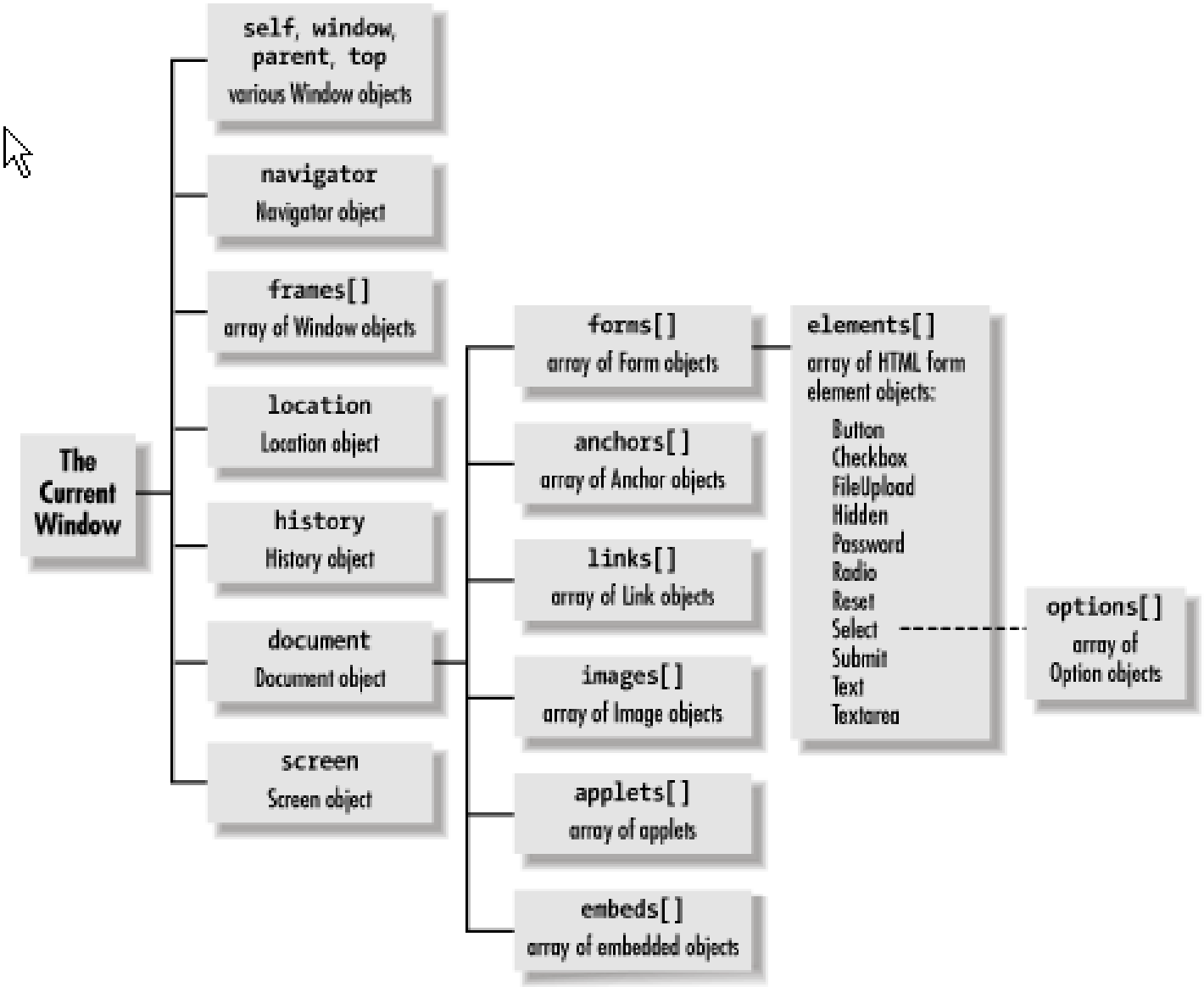
- `closed`
- `document`
- `frames`
- `history`
- `opener`
- `parent`
- `status`
- …

Objekt `Window`

Meetodid:

- `alert( ), confirm( ), prompt( )`
- `close( ), focus( )`
- `moveBy( ) , moveTo( )`
- `open( )`
- `resizeBy( ) , resizeTo( )`
- …

Objekt `Window`

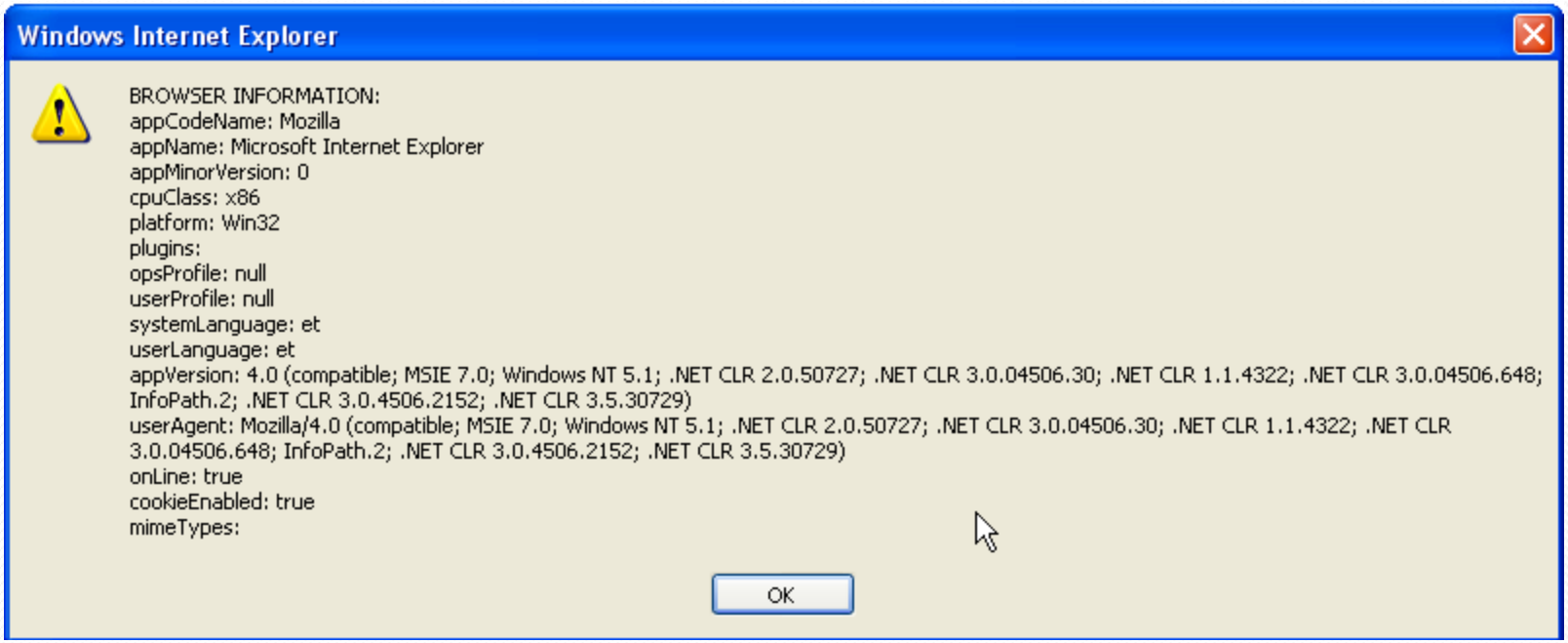


Objekt `Window`

```
window.answer = 42;

window.document.forms[0]

parent.frames[0].document.forms[0].elements[3].options[2].text
```

Objekt Navigator

```
var browser = "BROWSER INFORMATION:\n";
for(var propname in navigator) {
  browser += propname + ": " + navigator[propname] + "\n”}
alert(browser);
```



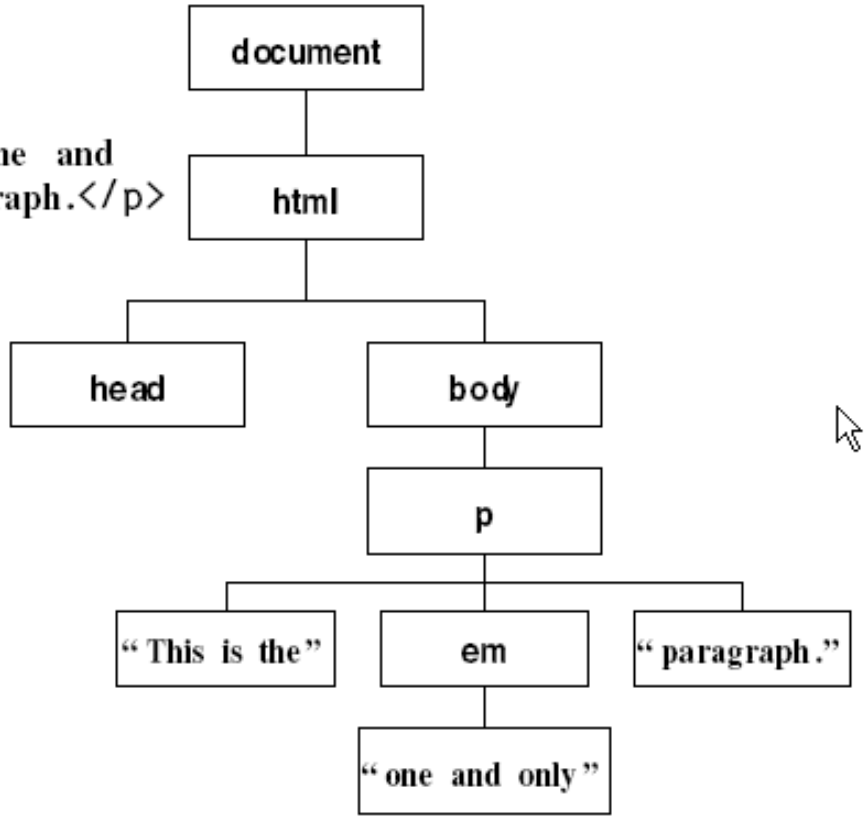
Objekt `History`

- Turvalisuse kaalutlustel ei ole JavaScriptist võimalik kasutada massiivi viimati külastatud lehtedega

- Kasutada saab `back()` ja `forward()` meetodeid

Objekt Document

```
<html>
  <head></head>
  <body>
      <p>This is the <em>one and
          only</em> paragraph.</p>
  </body>
</html>
```



Objekt `Document`

Meetodid:

- `write( )`
- `close( )`
- `open( )`
- `writeln( )`
- …

Objekt `Document`

Omadused:

- `bgColor, fgColor`
- `forms[]`
- `images[]`
- `lastModified`
- `links[]`
- …

Objekt Document

```
<form name="f1">
  <input type="button" value="Push Me">
</form>
```

Viitamine:

```
document.forms[0] //Oletame, et on esimene
document.f1       //Nime järgi
document.forms.f1 // Omaduse järgi
document.forms["f1"] // Massiivi järgi
```

Vormi elementide omadused

`type` - button, checkbox, radio, ..

`form` - read-only viit `Form` objektile, mille sees ta on

`name` - string

`value` – sõltub elemendist; string, mis saadetakse serverile peale submit()

Sündmuste töötlemine

Atribuut	Elemendid
onload	body, frameset
onunload	body, frameset
onclick	
ondblclick	UI elements
onmousedown	
onmouseup	
onmouseover	
onmouseout	
onmousemove	
onfocus	label, input, select, textarea, button
onkeypress	UI elements
onkeydown	
onkeyup	
onsubmit	form
onreset	
onselect	input, textarea
onchange	

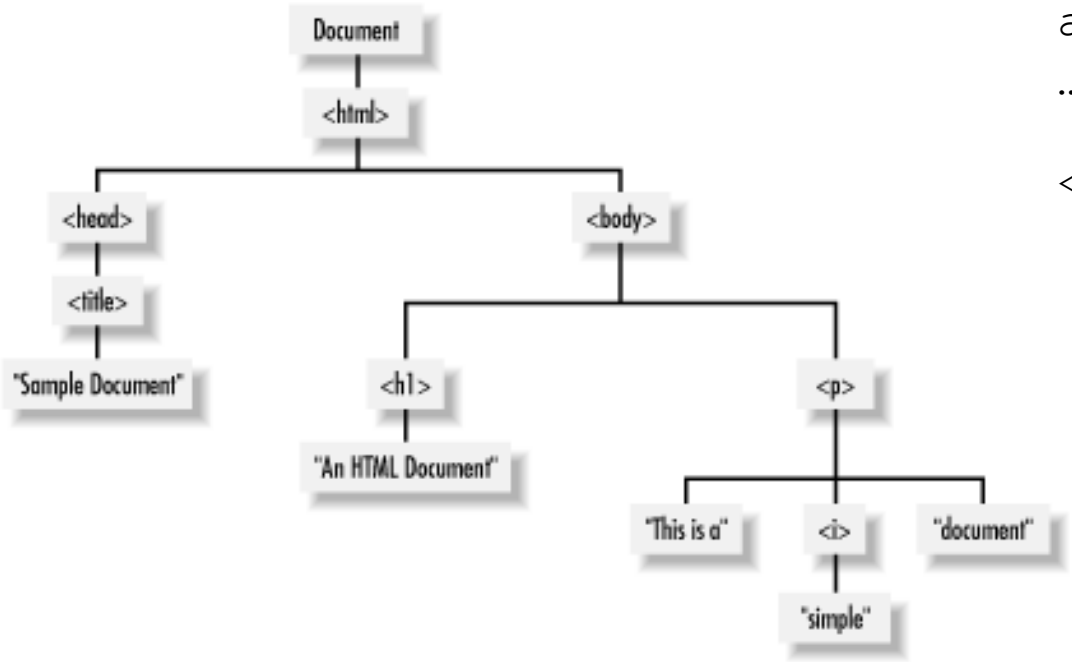
Iga kord kui kasutada sündmuste atribuute, tuleb dokumendi `head` elementi panna märgis `meta`:

```
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript" />
</head>
```

```
<body onload="window.alert('Body loaded.');"
      onunload="window.alert('Unloading...');">
    <form action="http://www.example.org"
          onsubmit="window.alert('Submitting...');"
          onreset="window.alert('Resetting...');">
          <input type="text" name="someText"
                onkeypress="window.alert('Text field got
                     character.');"
                onselect="window.alert('Text selected.');" />
</body>
```

Objekt Node

DOM defineerib üldise objekti Node, millel on meetodid ja omadused puu läbimiseks ja manipuleerimiseks



Node omadused:

```
nodeType
nodeName
parentNode
childNodes
previousSibling
nextSibling
attributes
…
```

```
<html>
  <head>
  <title>Sample
        Document</title>
  </head>
  <body> <h1>An HTML
        Document</h1>
    <p>This is a <i>simple</i>
        document.
  </body>
</html>
```

```
<html>
    <head>
        <title>A</title>
    </head>
    <body>
        <h1>HH1</h1>
        <p>Hello world!</p>
    </body>
</html>
```


```
alert( document.body.firstChild.innerHTML) tulemus  - HH1

alert( document.body.firstChild.nextSibling.innerHTML)

Tulemus -  Hello world!
```

Tippude tüübid

`nodeType` **väärtus**	**Tekstikonstant**	**Hosti objekti tüüp**
1	`Node.ELEMENT_NODE`	`Element`
2	`Node.ATTRIBUTE_NODE`	`Attr`
3	`Node.TEXT_NODE`	`Text`
8	`Node.COMMENT_NODE`	`Comment`
9	`Node.DOCUMENT_NODE`	`Document`
10	`Node.DOCUMENT_TYPE_NODE`	`DocumentType`

Objekt `Node`

Meetodid:

`Node` **meetod**	**Funktsionaalsus**
`hasAttributes()`	Kas antud tipul on atribuute.
`hasChildNodes()`	Kas antud tipul on järglasi.
`appendChild(Node)`	Lisab tipule alluva lõppu.
`insertBefore(Node, Node)`	Lisab tipu (esimene argument) antud tipu alluvatele enne teist argumenti `Node`.
`removeChild(Node)`	Eemaldab etteantud `Node` antud tipu alluvate hulgast.
`replaceChild(Node, Node)`	Antud tipu alluvate hulgas asendatakse teine `Node` esimese `Node`-ga.

Dokumendi `html` elemendile vastava objekti saame objekti `document` omadusega `documentElement`:

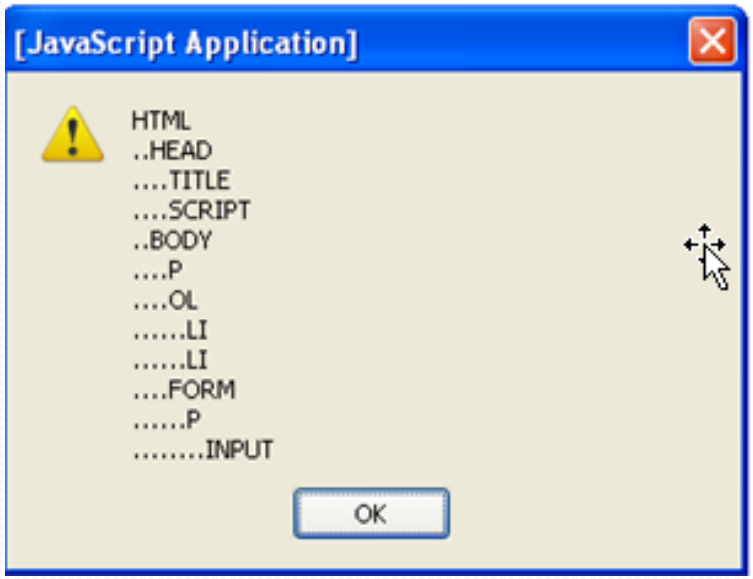
```
// TreeOutline.js
function treeOutline() {
  return subtreeOutline(document.documentElement, 0);
}
function subtreeOutline(root, level) {
  var retString = "";
  var elementType = window.Node ? Node.ELEMENT_NODE : 1;
  if (root.nodeType == elementType) {
    retString += printName(level, root.nodeName);
    var children = root.childNodes;
    for (var i=0; i<children.length; i++) {
      retString += subtreeOutline(children[i], level+1);
    }
  }
  return retString;
}
```

```
function printName(level, thisName) {
  var retString = "";
  for (var i=0; i<level; i++) {
    retString += "..";
  }
  retString += thisName + "\n";
  return retString;
}
```

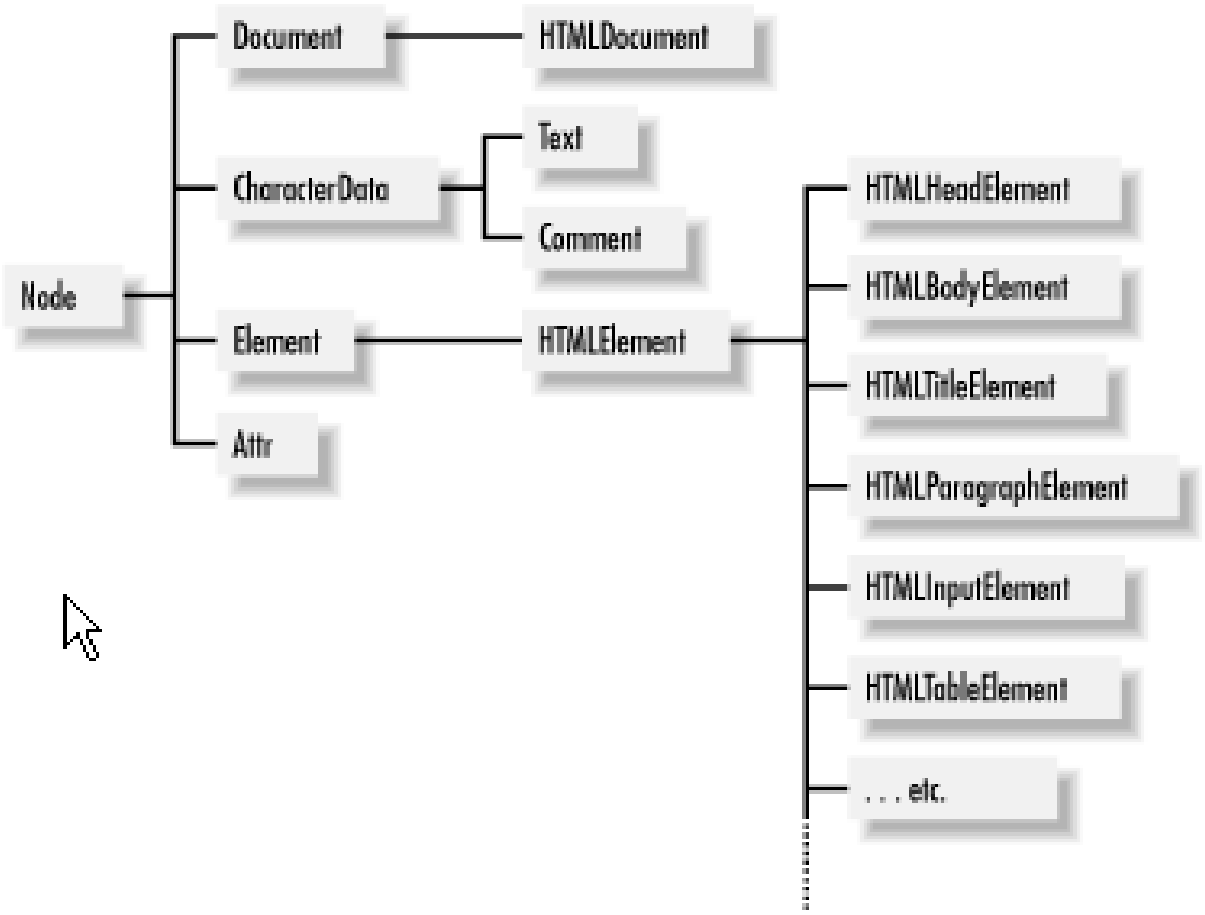
HTML dokument on järgmine:

```
<!DOCTYPE html
        PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>
      TreeOutline
    </title>
    <script type="text/javascript" src="TreeOutline.js">
    </script>
  </head>
  <body> <p>  Text within a "p" element. </p>
    <ol>
      <li>First element of ordered list.</li>
      <li>Second element.</li>
    </ol>
<form action="">
      <p><input type="button" name="button" value="Click to see
        outline”  onclick="window.alert(treeOutline());" /></p>
    </form>
</body>
</html>
```

HTML dokumendile vastav struktuur:



Node hierarhia



Document Node meetodid

Omadus	Väärtus
`domain`	String, mis esitab URL domeeni osa.
`referrer`	Kui antud dokument laaditi, sest klikiti linki, siis String sisaldab linki.
`createElement (String)`	Argumendiks on elemendi tüübi nimi (nt `div`). Tagastab objekti `Element` isendi, mis vastab sellele tüübile.
`createTextNode (String)`	Tagastab objekti `Text` isendi, mis sisaldab antud teksti.
`createElementById (String)`	Tagastab objekti `Element` isendi, mille `id` sisaldub argumendis.
`getElementsByTagName (String)`	Antud String on elemendi tüübi nimi. Tagastab objekti Element isendite massiivi, mille tüüp sisaldub argumendis.
`getElementById (String)`	Antud String on elemendi id väärtus. Tagastab objekti Element kui element eksisteerib.

Element Node

Unikaalne omadus – `tagName`

Meetod	**Tulemus**
`getAttribute(String)`	Atribuudi väärtus.
`setAttribute(String, String)`	Seatakse näidatud nimega atribuudi väärtus (või luuakse).
`removeAttribute(String)`	Eemaldatakse näidatud nimega atribuut.
`hasAttribute(String)`	Tagastab boolean.
`getElementsByTagName(String)`	Antud elemendi järglased (descendants).

Text Node

Põhiliseks omaduseks on `data`, mis esindab elemendis olevat teksti.

Brauseri toetuse kontrollimine

```
if (document.implementation &&
        document.implementation.hasFeature &&
        document.implementation.hasFeature("html", "1.0"))
{
// The browser claims to support Level 1 Core and HTML
//interfaces
}
```