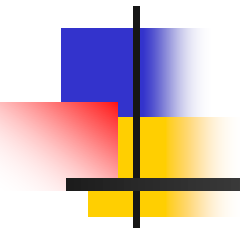
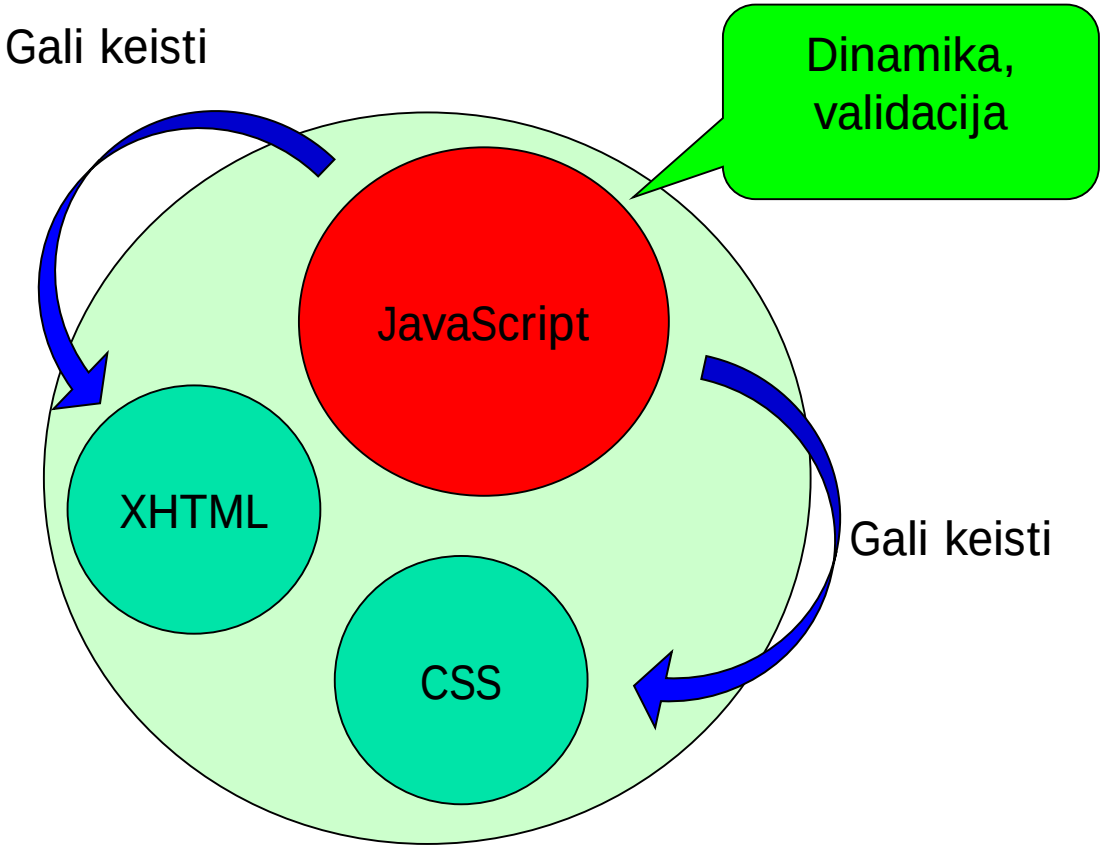


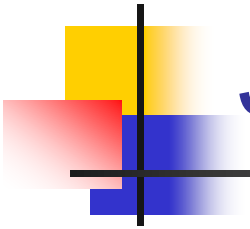
Interneto technologijos

JavaScript

Įvadas

JavaScript – dinaminių puslapių kūrimo priemonė





JavaScript

- *Skripto* kalba, sukurta Netscape kompanijos
  - Visos skripto kalbos paprastai yra interpretuojamos ir dinaminės (t.y., tipų surišimas yra dinaminis, kintamieji gali keisti savo tipą programos vykdymo metu)
- Tai ne Java!
  - Skirtumų tarp JavaScript ir Java yra daugiau nei panašumų
  - Java nėra nei interpretuojama, nei dinaminė kalba

JavaScript kalbos versijos

- JavaScript 1.5 (standartizuota kaip ECMA-262, trečia redakcija)
  - Plačiausiai palaikoma naršyklių
- 2011 birželį išėjo 5.1 ECMA-262 standarto redakcija:
  - http://www.ecma-international.org/publications/standards/Ecma-262.htm

JavaScript kalbos versijos

Version	Release date	Equivalent to	Netscape Navigator	Mozilla Firefox	Internet Explorer
1	March 1996		2		3
1.1	August 1996		3		
1.2	June 1997		4.0-4.05		
1.3	October 1998	ECMA-262 1st edition / ECMA-262 2nd edition	4.06-4.7x		4
1.4			Netscape Server		
1.5	**November 2000**	**ECMA-262 3rd edition**	**6**	**1**	**5.5 - 8**
1.6	November 2005	1.5 + Array extras + Array and String generics + E4X		1.5	
1.7	October 2006	1.6 + Pythonic generators + Iterators + let		2	
1.8	June 2008	1.7 + Generator expressions + Expression closures		3	
1.8.1		1.8 + Minor Updates		3.5	
1.8.5		1.8.1 + ECMAScript 5 Compliance		4	

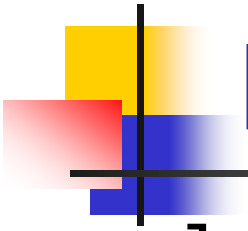
JavaScript galimybės

- Kas įmanoma su JavaScript
  - Bendravimas su vartotoju (pranešimai, dialogai)
  - HTML dokumento **turinio** dinaminis keitimas
  - HTML dokumento **stiliaus** dinaminis keitimas
  - Bendravimas su HTML puslapyje esančiais komponentais (Applets, ActiveX)
- Ko negalima daryti su JavaScript
  - Negalima dirbti su failais (skaityti/rašyti)
  - Negalima bendrauti su kitais kompiuteriais per tinklą (išskyrus serverį, iš kurio atėjo HTML puslapis)

JavaScript pavyzdys

- HTML dokumento turinio dinaminio formavimo pavyzdys

```
<html>
<head>...</head>
<body>
<script type="text/javascript">
    document.write("<h2>Table of Factorials</h2>");
    for(var i=1, var fact=1; i<10; i++, fact*=i) {
        document.write(i + "! = " + fact);
        document.write("<br>");
    }
</script>
</body>
</html>
```



Duomenų tipai

- JavaScript turi šiuos duomenų tipus:
  - Skaičiai: 42 arba 3.14159
  - Loginės (Boolean) reikšmės: `true` arba `false`
  - Eilutės: "Kuku"
  - `null` reikšmė (ši reikšmė yra be tipo)
    - **JavaScript yra *case sensitive* kalba**, null yra ne tas pats kaip Null, NULL, ar panašiai.
  - `undefined` reikšmė: žymi, kad kintamasis yra neinicializuotas
  - Objektai (JavaScript turi kai kurių OO kalbų savybių!), pvz., galima sukurti datos objektą
    - Masyvai yra objektai!
  - Funkcijos (funkciją galima priskirti kintamajam ir/arba perduoti kaip parametrą)

Automatinė tipų konversija

- JavaScript yra dinamiškai tipizuota kalba (t.y., tipai kintamiesiems priskiriami tik programos vykdymo metu)
- Kintamųjų paskelbimo metu tipų nurodyti nereikia, pvz.:
  - `var atsakymas = 42`
- Vėliau tam pačiam kintamajam galima priskirto kito tipo reikšmę:
  - `atsakymas = "Tam param tatam..."`
  - kintamojo tipas keičiamas *dinamiškai*, jokios klaidos nebus!
- Aritmetiniai operatoriai skirtingų tipų reikšmes konvertuoja į vieno tipo reikšmes automatiškai
  - Sudėties operatorius, jei bent vienas operandas nėra skaičius, operandus verčia į eilutės tipo reikšmes:
    - ```
      x = "Ats.: " + 42 // reikšmė: "Ats.: 42"
      y = 42 + " Lt." // reikšmė: "42 Lt."
      ```
  - Kiti operatoriai bando eilutės tipo reikšmes versti į skaitines:
    - ```
      "37" - 7 // 30
      "37" + 7 // 377
      ```

Kintamieji

- Kintamąjį paskelbti galima dviem būdais:
  - Naudojant raktinį žodį `var`:
    - `var x = 42`
  - Nenaudojant var:
    - `x = 42`
- Neinicializuotų kintamųjų naudojimas:
  - Jei kintamasis buvo paskelbtas be `var`, bus klaida

```
x;
y = x+1; // klaida
```

  - Jei su `var`, skaitiniame kontekste jo reikšmė bus `NaN`, kituose - `undefined`

```
var x;
y = x+1; // NaN
```

`undefined` ir `null` reikšmės

- Yra galimybė patikrinti, ar kintamasis buvo inicializuotas:

```
var input;
if(input === undefined){
   doThis();
} else {
   doThat();
}
```

- `undefined` loginėse išraiškose yra verčiama į `false`:

```
function suma(a, b) {
  if (a && b)
    return a+b;
    else return 0;
}
```

- `null` reikšmė elgiasi kaip nulis skaitinėse išraiškose, ir kaip `false` loginėse išraiškose:

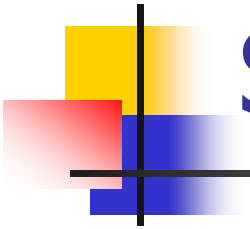
```
var n = null
n * 32 // bus 0
```

Kintamųjų galiojimo sritys

- Kintamieji, paskelbti funkcijų viduje, yra *lokalūs* kintamieji, visi kiti yra *globalūs* kintamieji
- Globalius kintamuosius galima paskelbti ir su `var`, ir be `var`, lokalius – tik su `var`

```
var x; // globalus
y;     // globalus

function suma(a, b) {
  var s = a+b; // s yra lokalus
  return s;
}
```



Skaitinės ir loginės konstantos

- JavaScript neskiria sveikų skaičių nuo realių (kol kintamojo reikšmė sveika, kablelis nespausdinamas)
- Šešioliktainiai skaičiai gali būti užrašyti, pridedant prefiksą `0x`:
  - `0xFFFF`
- Yra dvi loginės konstantos:
  - `true`
  - `false`

Konstantos-eilutės (string)

- Apskliaudžiamos kabutėmis arba apostrofais:

```
"kuku"
'aha'
"Katinas pasakė 'miau'"
"pirma eilutė \n antra eilutė"
```

- "\" naudojamas kaip *escape* simbolis

Escape simboliai

Character	Meaning
\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	Tab
\'	Apostrophe or single quote
\"	Double quote
\\	Backslash character (\)
XXX	The character with the Latin-1 encoding specified by up to three octal digits *XXX* between 0 and 377. For example, \251 is the octal sequence for the copyright symbol.
\x*XX*	The character with the Latin-1 encoding specified by the two hexadecimal digits *XX* between 00 and FF. For example, \xA9 is the hexadecimal sequence for the copyright symbol.
\u*XXXX*	The Unicode character specified by the four hexadecimal digits *XXXX*. For example, \u00A9 is the Unicode sequence for the copyright symbol.

Masyvai-konstantos

- Masyvai-konstantos paskelbiami su `[]`, numeruojami nuo 0. Pvz.: trijų elementų masyvas:
  - `var spalvos = ["juoda", "balta", "raudona"];`
- Kai kuriuos masyvo elementus galima praleisti:
  - `var spalvos = ["juoda", , "balta", "raudona"];`
  - `spalvos[0]` – "juoda", `spalvos[1]` – `undefined`
- Masyvo elementai gali būti skirtingų tipų!
  - `var strange = ["abc", 14, true, new Date()];`

Objektai-konstantos

- Objektą-konstantą paskelbti galime naudodami riestinius skliaustelius `{}`, kurių viduje rašomos *savybės* ir jų *reikšmės*:

```
var automobilis = {marke: "Audi",
        turis: 2.0, spalva: "raudona"}
alert(automobilis.marke);
```

Priskyrimo operatorius "="

Shorthand operator	Meaning
x += y	x = x + y
x -= y	x = x - y
x *= y	x = x * y
x /= y	x = x / y
x %= y	x = x % y
x <<= y	x = x << y
x >>= y	x = x >> y
x >>>= y	x = x >>> y
x &= y	x = x & y
x ^= y	x = x ^ y
x |= y	x = x | y

Palyginimo operatoriai

Operator	Description	Examples returning true[1]

Equal (==)	Returns true if the operands are equal. If the two operands are not of the same type, JavaScript attempts to convert the operands to an appropriate type for the comparison.	`3 == var1`
		`"3" == var1`
		`3 == '3'`

Not equal (!=)	Returns true if the operands are not equal. If the two operands are not of the same type, JavaScript attempts to convert the operands to an appropriate type for the comparison.	`var1 != 4` `var2 != "3"`
Strict equal (===)	Returns true if the operands are equal and of the same type.	`3 === var1`
Strict not equal (!==)	Returns true if the operands are not equal and/or not of the same type.	`var1 !== "3"` `3 !== '3'`
Greater than (>)	Returns true if the left operand is greater than the right operand.	`var2 > var1`
Greater than or equal (>=)	Returns true if the left operand is greater than or equal to the right operand.	`var2 >= var1` `var1 >= 3`
Less than (<)	Returns true if the left operand is less than the right operand.	`var1 < var2`
Less than or equal (<=)	Returns true if the left operand is less than or equal to the right operand.	`var1 <= var2` `var2 <= 5`

[1] These examples assume that `var1` has been assigned the value 3 and `var2` has been assigned the value 4.

Aritmetiniai operatoriai

- Įprasti `+`, `-`, `*`, `/`
  - `/` visada atlieka **realių** skaičių dalybą (net jei abu operandai sveiki skaičiai)
  - ```
    1/2  // returns 0.5 in JavaScript
    1/2  // returns 0 in Java
    ```

Operator	Description	Example
% (Modulus)	Binary operator. Returns the integer remainder of dividing the two operands.	12 % 5 returns 2.
++ (Increment)	Unary operator. Adds one to its operand. If used as a prefix operator (++x), returns the value of its operand after adding one; if used as a postfix operator (x++), returns the value of its operand before adding one.	If x is 3, then ++x sets x to 4 and returns 4, whereas x++ sets x to 4 and returns 3.
-- (Decrement)	Unary operator. Subtracts one to its operand. The return value is analogous to that for the increment operator.	If x is 3, then --x sets x to 2 and returns 2, whereas x-- sets x to 2 and returns 3.
    ```

Loginiai operatoriai

Operator	Usage	Description
&&	`expr1 && expr2`	(Logical AND) Returns `expr1` if it can be converted to false; otherwise, returns `expr2`. Thus, when used with Boolean values, && returns true if both operands are true; otherwise, returns false.
||	`expr1 || expr2`	(Logical OR) Returns `expr1` if it can be converted to true; otherwise, returns `expr2`. Thus, when used with Boolean values, || returns true if either operand is true; if both are false, returns false.
!	`!expr`	(Logical NOT) Returns false if its single operand can be converted to true; otherwise, returns true.

Specialūs operatoriai

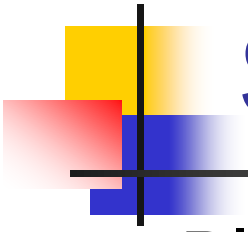
- `new` – skirtas kurti objektus

```
var date = new Date();
```

- `this` – skirtas prieiti prie einamojo objekto

```
<input type="text" name="amzius"
       onchange="validate(this)"/>

function validate(obj) {
  if ((obj.value < 0) || (obj.value > 200))
    alert("Bloga reikšmė!")
}
```



Sakiniai

- Blokai `if-then-else`, `switch`, `for`, `while`, `do-while` tokie patys kaip ir Java/C++ kalbose
- Sakiniai baigiami kabliataškiu
- (!) `for` ciklo kintamuosius reikia paskelbti su `var`, ne su `int`

```
for(var i=1, var fact=1; i<10; i++, fact*=i) {
    document.write(i + "! = " + fact);
    document.write("<br>");
}
```

Funkcijos

- Paskelbiamos su raktiniu žodžiu `function`
- Gali grąžinti rezultatą, gali ir negrąžinti

```
function suma(a, b) {
  if (a && b)
    return a+b; // else atveju nieko negrąžiname
}
```

- Funkciją kviesti galima su bet kokiu parametrų skaičiumi
  - `suma(1, 2); suma(1); suma();`
    `suma("lala", 14, true);`

Funkcijos argumentų masyvas

- Visi funkcijai paduoti argumentai yra saugomi masyve `arguments[]`
- Pavyzdys: funkcija, sujungianti visus argumentus į eilutę, naudodama pirmą argumentą kaip skirtuką:

```
function myConcat(separator) {
   result="";
   // praleidžiam pirmą argumentą (skirtuką):
   for (var i=1; i<arguments.length; i++) {
      result += arguments[i] + separator;
   }
   return result;
}
```

- Galime šia funkciją kviesti taip:
  - `myConcat(":", "AB", "CD", "EF");`

Standartinės funkcijos

- `parseInt(String [, radix])` – verčia eilutę į skaičių pagal nurodytą skaičiavimo sistemą (dešimtainė pagal nutylėjimą)
  - `parseInt("123") === 123`
  - `parseInt("12abc") === 12`
  - `parseInt("a", 16) === 10`
  - `parseInt("a")` – grąžins `NaN`
- Kaip patikrinti, ar kintamojo reikšmė sveikas skaičius:
  - `parseInt(x) == x`
- `isNaN(value)` – tikrina, ar paduotas argumentas yra reikšmė NaN

Taikymai: datos validavimas

```
function checkDate(yearStr, monthStr, dayStr)
    if (yearStr != parseInt(yearStr)) {
        return false;
    }
    ...

    year = parseInt(yearStr);
    month = parseInt(monthStr)-1; // Sausis - 0
    day = parseInt(dayStr);
    if (month < 0 || month > 11) {
        ...
    }
    var date = new Date(year, month, day);
    if (date.getDate() != day) {
        alert("Neteisinga data");
        return false;
    }
}
```

Įvykių apdorojimas

- Pavyzdys:

```
<form name="manoforma" action="...">
...
    <input type="button" value="Paspausk"
           onClick="funk()"/>
...
</form>
```

- Kiti įvykiai:
  - onChange
  - onFocus, onBlur
  - ...