

Тема 12. **Условный и циклический
оператор в JavaScript.
Регулярные выражения**

СОДЕРЖАНИЕ

1. Логические операторы и операторы сравнения в JS
2. Условные операторы в JS
3. Циклические операторы в JS
4. Регулярные выражения
5. Примеры



ЛОГИЧЕСКИЕ ОПЕРАТОРЫ И ОПЕРАТОРЫ СРАВНЕНИЯ В JS

- Логические операторы и операторы сравнения в JS возвращают одно из двух следующих значений: „**true**” или „**false**”
- Логические операторы используются для определения логики между переменными или значениями переменных
 - **&&** - and (и)
 - **||** - or (или)
 - **!** - not (отрицание)
- Операторы сравнения используются для определения равенства или различий между переменными или значениями переменных
 - **==** - равно
 - **===** - равны и значение и тип
 - **!=** - не равно
 - **!==** - не равно ни значение ни тип
 - **>** - больше чем
 - **<** - меньше чем
 - **>=** - больше или равно чем
 - ▶ **<=** - меньше или равно чем

ПРИМЕР ИСПОЛЬЗОВАНИЯ ОПЕРАТОРА „===”

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<p>Нажмите кнопку для визуализации равенства двух значений - 5 равно с  
"5" как значение и как тип?</p>
```

```
<button onclick="Comparatie()">Click</button>
```

```
<p id="et"></p>
```

```
<script>
```

```
function Comparatie() {
```

```
    var x = 5;
```

```
    document.getElementById("et").innerHTML = (x === "5");
```

```
}
```

Нажмите кнопку для визуализации равенства двух значений - 5 равно с "5" как значение и как тип?

```
</script></body>
```

Click

```
</html>
```

false



УСЛОВНЫЙ ОПЕРАТОР В JS

□ JavaScript содержит, условный оператор который присваивает значение переменной, в зависимости от выполнения или не выполнения определенного условия

□ Синтаксис:

***названиеПеременной = (условие) ? значение1:
значение2***



ПРИМЕР ИСПОЛЬЗОВАНИЯ УСЛОВНОГО ОПЕРАТОРА JS

...<p>Выберите опцию определяющую ваш пол:</p>

```
<select id="aleg">
```

```
  <option value="barbat">Мужской</option>
```

```
  <option value="femeie">Женский</option>
```

```
</select>
```

```
<button onclick="Verificare()">Click</button>
```

```
<p id="et2"></p><script>
```

```
function Verificare() {
```

```
  var alegere,sex;
```

```
  alegere = document.getElementById("aleg").value;
```

```
  sex = (alegere == "barbat") ? " господин!!!":" госпожа!!!";
```

```
  document.getElementById("et2").innerHTML = "Хорошего вам дня" +  
sex;}
```

```
</script>...
```



РЕЗУЛЬТАТ ПРИМЕРА

Выберите опцию определяющую ваш пол:

Женский ▼

Click

Хорошего вам дня госпожа!!!

Выберите опцию определяющую ваш пол:


Мужской ▼

Click

Хорошего вам дня господин!!!



ДРУГИЕ УСЛОВНЫЕ ОПЕРАТОРЫ

- Используются, обычно, для выполнения разных действий, на основе разных условий
 - В JavaScript существует возможность использования следующих условных операторов:
 - **if** - используются когда необходимо выполнить блок JS-кодов, тогда когда заданное условие является верным (true)
 - **else** - используются когда необходимо выполнить блок JS-кодов, тогда когда заданное условие является неверным (false)
 - **else if** - используются для определения нового условия, в случае когда первое условие неверное
 - **switch** – используются для определения нескольких альтернативных блоков JS-кодов, которые могли бы выполняться при выполнении соответствующего условия
- 

ОПЕРАТОР „IF”

- Оператор „**IF**”, форма условного оператора, и используется тогда когда необходимо выполнить блок JS-кодов, при верным заданном условии
- Основная форма:

if (условие) {

блок JS-кодов необходимый выполнить при выполнении условия

}

Задача:

Утро считается тогда когда время не превышает 10 часов дня.



ОБЪЕКТ „DATE()”

- Используется когда необходимо работать с датой и временем
- Объекты типа „дата” создаются **new Date()**
- Некоторые методы объекта „**Date**”:

Метод	Описание
getDate()	Возвращает день месяца (число между 1-31)
getDay()	Возвращает день недели (число между 0-6)
getFullYear()	Возвращает год (в цифрах)
getHours()	Возвращает час (число между 0-23)
getMilliseconds()	Возвращает миллисекунды (между 0-999)
getMinutes()	Возвращает минуты (между 0-59)
getMonth()	Возвращает месяц (между 0-11)
getSeconds()	Возвращает секунды (число между 0-59)

Прим: Аналогично, существуют методы используемые для установки деталей даты

ПРИМЕР ИСПОЛЬЗОВАНИЯ ОПЕРАТОРА

„IF”

... <p>Выберите опцию определяющую ваш пол:</p>

```
<select id="aleg">
```

```
  <option value="barbat">Мужской</option>
```

```
  <option value="femeie">Женский</option>
```

```
</select>
```

```
<button onclick="Verificare()">Click</button>
```

```
<p id="et2"></p><p id="etZi"></p>
```

```
<script>
```

```
function Verificare() {
```

```
  var alegere,sex;
```

```
  alegere = document.getElementById("aleg").value;
```

```
  sex = (alegere == "barbat") ? " господин!!!":" госпожа!!!";
```

```
  var timp = new Date().getHours();
```

```
  if (timp < 10) {
```

```
    document.getElementById("et2").innerHTML = "Хорошего вам утра " + sex;}  
}
```

}....

ФОРМА „IF...ELSE” ОПЕРАТОРА „IF”

- „**Else**” используется когда необходимо выполнение блока JS-кодов в случае когда условие не выполняется

```
... if (timp < 10) {
```

```
    document.getElementById("et2").innerHTML = "Хорошего вам  
утра " + sex;
```

```
    } else if (timp < 17) {
```

```
        document.getElementById("et2").innerHTML = "Хорошего вам  
дня " + sex;
```

```
}
```

Выберите опцию определяющую ваш пол:

Хорошего вам дня госпожа!!!



ФОРМА „IF...ELSE IF” ОПЕРАТОРА IF

- Используется для установки нового условия, в случае когда первое условие не верное

```
... var timp = new Date().getHours();
```

```
if (timp < 10) {
```

```
    document.getElementById("et2").innerHTML = "Хорошего вам  
утра " + sex;
```

```
} else if (timp < 17) {
```

```
    document.getElementById("et2").innerHTML = "Хорошего вам  
дня " + sex;
```

```
} else {document.getElementById("et2").innerHTML = "Хорошего  
вам вечера " + sex;;} ...
```

Выберите опцию определяющую ваш пол:

Женский ▾

Click

Хорошего вам вечера госпожа!!!



ОПЕРАТОР „SWITCH”

- Используется для выполнения разных действий в зависимости от выполнения определенных условий
- Основная форма:

```
switch(выражение) {  
    case n1:  
        блок с кодами;  
        break;  
    case n2:  
        блок с кодами;  
        break;  
    ...  
    default:  
        блок с кодами (по умолчанию);
```



где:

- „**выражение**” – вычисляется один раз
- Значение выражения сравнивается с каждым значением случаев
- При нахождении соответствия значений, соответствующий блок будет выполнен
- При встрече резервированного слова „**break**” происходит выход из блока случая
- Блок соответствующий резервированному слову „**default**” выполнится в случае когда не найдется ни одно соответствие между сравнениями

ПРИМЕР SWITCH

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<p>Выберите опцию определяющую ваш пол:</p>
```

```
<select id="aleg">
```

```
  <option value="barbat">Мужской</option>
```

```
  <option value="femeie">Женский</option>
```

```
</select>
```

```
<button onclick="Verificare()">Click</button>
```

```
<p id="et2"></p>
```

```
<p id="etZi"></p>
```



ПРИМЕР SWITCH, продолжение

```
<script>
```

```
function Verificare() {  
    var alegere,sex;  
    alegere = document.getElementById("aleg").value;  
    sex = (alegere == "barbat") ? " господин!!!" : " госпожа!!!";  
    var timp = new Date().getHours();  
    if (timp < 10) {  
        document.getElementById("et2").innerHTML = "Хорошего вам утра " +  
sex;  
    } else if (timp < 17) {  
        document.getElementById("et2").innerHTML = "Хорошего вам дня " + sex;  
    } else {document.getElementById("et2").innerHTML = "Хорошего вам вечера "  
+ sex;  
    }  
}
```



ПРИМЕР SWITCH, продолжение

```
var zi;
switch (new Date().getDay()) {
    case 0:
        zi = "Воскресенье"; break;
    case 1:
        zi = "Понедельник"; break;
    case 2:
        zi = "Вторник"; break;
    case 3:
        zi = "Среда"; break;
    case 4:
        zi = "Четверг"; break;
    case 5:
        zi = "Пятница"; break;
    case 6:
        zi = "Субота"; break;
}
```

```
document.getElementById("etZi").innerHTML
= "Сегодня - " + zi + " " + new
Date().getDate() + "." + new
Date().getMonth()
+ ", а текущее время - " + new
Date().getHours() + ":" + new
Date().getMinutes();
}
</script>
</body>
</html>
```



}

Вывод времени с обновлением

Fișierul .html

```
...<head>
<script src="ora.js">
</script>
</head>
<body
onload="afisare()">
  <p id="etZi"></p>...
</body>
```

Fișierul ora.js

```
setInterval(function()
{afisare()},30000);
function afisare()
```

```
switch (new Date().getDay()) {
  case 0: zi = "Duminica"; break;
  case 1: zi = "Luni"; break;
  case 2: zi = "Marti"; break;
  case 3: zi = "Miercuri"; break;
  case 4: zi = "Joi"; break;
  case 5: zi = "Vineri"; break;
  case 6: zi = "Sambata"; break;}
document.getElementById("etZi").inner
HTML = "Astazi este " + zi + " " + new
Date().getDate() + "." + new
Date().getMonth() + ", iar ora curenta
este " + new Date().getHours() + ":" +
new Date().getMinutes();
}
```

{

РЕЗУЛЬТАТ ПРИМЕРА

Выберите опцию определяющую ваш пол:

▼

Хорошего вам дня госпожа!!!

Сегодня - Понедельник 17.10, а текущее время - 14:14



ЦИКЛИЧЕСКИЙ ОПЕРАТОР

- Циклический оператор используется для повторного выполнения, блока JS-кодов
- JavaScript имеет несколько циклических операторов:
 - **for** - повторяет блок кодов несколько раз
 - **for/in** - повторяет действия относящиеся к свойствам определенного объекта
 - **while** - повторяет блок кодов, до тех пор пока удовлетворяется условие
 - **do/while** - повторяет блок кодов в то время как условие удовлетворяется



ОПЕРАТОР „FOR”

- Используется тогда когда необходимо повторное выполнение блока кодов
- Основная форма:

```
for (выр 1; выр 2; выр 3) {  
    блок кодов необходимых для выполнения;  
}
```

где

- *выр 1* – определяет начальное значение индикатора цикла;
- *выр 2* – условие выполнения цикла;
- *выр 3* – выполняется после выполнения блока кодов цикла (шаг, инкремент)

Прим: Оператор „+=” используется для конкатенации строк



ПРИМЕР С ОПЕРАТОРОМ „FOR”

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<p>Нажмите кнопку чтобы увидеть эффект оператора for</p>
```

```
<button onclick="functieFor()">Click</button>
```

```
<p id="etCiclu"></p><script>
```

```
<script>
```

```
function functieFor() {
```

```
    var text = "";
```

```
    var i;
```

```
    for (i = 0; i <= 3; i++) {
```

```
        text += "Выводится " + i + "-е число <br />";
```

```
    }
```

```
    document.getElementById("etCiclu").innerHTML = text;}</script></body></html>
```

Нажмите кнопку чтобы увидеть эффект оператора **for**

Click

Выводится 0-е число

Выводится 1-е число

Выводится 2-е число

Выводится 3-е число



ПРИМЕР „FOR..IN”

□ Данная форма используется для прохождения по свойствам определенного объекта

□ Пример:

...<p>Нажмите кнопку чтобы увидеть эффект оператора **for..in**</p>

<button onclick="functieForIn()">Click</button>

<p id="etCicluln"></p>

<script>

function functieForIn() {

var txt = "";

var persoana = {nume:"Иванов", prenume:"Иван", varsta:22};

var x;

for (x in persoana) {

txt += persoana[x] + " ";

}

document.getElementById("etCicluln").innerHTML = txt;}

▶</script>

РЕЗУЛЬТАТ ПРИМЕРА „FOR..IN”

Нажмите кнопку чтобы увидеть эффект оператора **for..in**

Click

Иванов Иван 22



ОПЕРАТОР „WHILE”

- Выполняет блок кодов, до тех пор пока удовлетворяется условие
- Основная форма:

while (условие) {

блок кодов необходимых для выполнения;

}



ПРИМЕР „WHILE”

```
...<script>
function functieWhile() {
    var text = "";
    var i = 0;
    while (i <= 3) {
        text += "Выводится " + i + "-е число <br />";
        i++;
    }
    document.getElementById("etCiclu").innerHTML =
text;
}
</script>...
```

Нажмите кнопку чтобы увидеть эффект оператора **while**

Click

Прим: не забудьте
инкрементировать переменную
цикла, в противном случае
действие браузера заблокируется

Выводится 0-е число
Выводится 1-е число
Выводится 2-е число
Выводится 3-е число



ЦИКЛ „DO..WHILE”

- Этот цикл похож на „**while**”, отличие в том что коды необходимые повторить выполняются минимум один раз, до того как проверится условие
- Выполнение блока осуществится повторно до тех пор пока условие верное
- Основная форма:

do {

***блок кодов необходимых для
выполнения;***

}

while (условие);



ПРИМЕР „DO..WHILE”

```
<script>
```

```
function funcieVWhile() {
```

```
    var text = "";
```

```
    var i = 0;
```

```
    do {
```

```
        text += "Выводится " + i + "-е число <br />";
```

```
        i++;
```

```
    }
```

```
    while (i <= 3);
```

```
    document.getElementById("etCiclu").innerHTML = text;
```

```
}
```

```
</script>...
```

Нажмите кнопку чтобы увидеть эффект оператора **while**

Click

Выводится 0-е число

Выводится 1-е число

Выводится 2-е число

Выводится 3-е число



РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ В JS

- Регулярное выражение представляет собой набор символов, которые определяют шаблон поиска
- Оно используется для
 - Поиска текста удовлетворяющего шаблон
 - Замены найденного текста другим текстом
- Регулярное выражение может быть сформировано из одного или более символов
- Основная форма: ***/pattern/*модификаторы;**
- Модификаторы:
 - ▣ ***i*** – реализует case-независимый поиск
 - ▣ ***g*** – реализует глобальный поиск (производится поиск всех совпадений)
- Пример: **/покрывало/i**
 - ▣ где **покрывало** - шаблон, а ***i*** - модификатор, указывающий чтобы поиск производился вне зависимости от регистра



ШАБЛОНЫ В РЕГУЛЯРНЫХ ВЫРАЖЕНИЯХ

Выражение	Описание
[abc]	Ищет любой указанный символ между скобками
[^abc]	Ищет символ, отличающийся от тех которые между скобками
[0-9]	Ищет цифру из указанного промежутка
(x y)	Ищет один из тех символов указанных через разделитель
[^0-7]	Ищет цифры кроме тех указанных между скобками

Прим: Тогда когда необходимо использовать специальные символы: (,), [,], {, }, + и др.. в шаблон регулярного выражения используется \ (back slash) перед символом



ШАБЛОНЫ В РЕГУЛЯРНЫХ ВЫРАЖЕНИЯХ. МЕТАСИМВОЛЫ

Метасимволы	Описание
<code>.</code>	Ищет только один символ
<code>\w</code>	Ищет совпадение слов
<code>\W</code>	Ищет не словесные совпадения
<code>\n</code>	Ищет символ «новая строка»
<code>\t</code>	Ищет символ „tab”
<code>\d</code>	Ищет только цифры
<code>\s</code>	Ищет символ «пробел»
<code>\b</code>	Ищет совпадения в начале или в конце слова
<code>\uxxxx</code>	Ищет unіcod-символ соответствующий шестнадцатеричному числу xxxx



КВАНТИФИКАТОРЫ В ШАБЛОНАХ

Quantificatori	Descrerie
n+	Ищет любую последовательность содержащую минимум один n
n*	Ищет любую последовательность содержащую ноль или несколько появлений n
n?	Ищет любую последовательность содержащую ноль или появление n
n{X}	Ищет любую последовательность содержащую последовательность состоящая из n , X раз
n{X,Y}	Ищет любую последовательность содержащую последовательность состоящая из n , в количестве от X до Y
n{X,}	Ищет любую последовательность содержащую n минимум X раз
n\$	Ищет последовательности содержащие n в конце
^n	Ищет последовательности содержащие n в начале
?=n	Ищет последовательности за которыми следует n
?!n	Ищет последовательности за которыми не следует n

JS-МЕТОДЫ ИСПОЛЬЗУЕМЫЕ С РЕГУЛЯРНЫМИ ВЫРАЖЕНИЯМИ

- В JS регулярные выражения используют несколько методов для работы с символьной строкой:
 - **search()** – использует регулярное выражение для поиска подстроки, и возвращает позицию найденной подстроки
 - **replace()** – возвращает измененную строку применив заданный шаблон
 - **match()** – ищет последовательность совпадений, используя регулярное выражение и возвращает совпадения в виде вектора со значениями



ПРИМЕР ОПРЕДЕЛЕНИЯ И ИСПОЛЬЗОВАНИЯ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ

```
<!DOCTYPE html>
```

```
<head>
```

```
  <!--meta charset="utf-8" / -->   <title>Проверка полей</title>
```

```
  <link rel="stylesheet" type="text/css" href="regExp.css" />
```

```
</head>
```

```
<html>
```

```
<body>
```

```
<form onsubmit="return Verificare()" name = "datePers">
```

```
<fieldset><legend>Контактные данные</legend>
```

```
  <p>Введите вашу фамилию: </p>
```

```
  <input id="nume" type="text" name="nume" />
```

```
  <span id="err_nume" class="err"></span>
```

```
  <p class="standard">Фамилия должна содержать минимум 2 буквы - первая  
большая!!!</p><br />
```

```
  <p>Введите ваше имя: </p><input id="prenume" type="text" name="prenume" />
```

```
  <span id="err_prenume" class="err"></span>
```

```
  <p class="standard">Имя должно содержать минимум 2 буквы - первая  
большая!!!</p><br />
```

ПРИМЕР ОПРЕДЕЛЕНИЯ И ИСПОЛЬЗОВАНИЯ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ. Продолжение

```
<p>Введите ваш номер телефона: </p>
```

```
  <input id="telefon" type="text" name="telefon" />
```

```
  <span id="err_tel" class="err"></span>
```

```
  <p class="standard">Номер телефона начинается с '+' и потом следуют 11 цифр  
(+xxxxxxxxxxx)!!!</p><br />
```

```
  <br />
```

```
  <input type="reset" value="Обновить" />
```

```
  <input type="submit" value="Сохранить" />
```

```
</fieldset>
```

```
</form>
```

```
<script>
```

```
function Verificare()
```

```
{
```

```
var necazuri=0;
```

```
var numeC, prenumeC, telefonC;
```


```
var numeSablou, prenumeSablou, telefonSablou;
```



ПРИМЕР ОПРЕДЕЛЕНИЯ И ИСПОЛЬЗОВАНИЯ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ. Продолжение

```
document.getElementById("err_ume").innerHTML = "";
    umeC = document.getElementById("ume").value;
    umeSablon = /^[A-Z][a-z]+$/;
    if (!(umeC.match(umeSablon))) {
        document.getElementById("err_ume").innerHTML = "Фамилия должна
содержать минимум 2 буквы - первая большая!!!";
        necazuri++;
    }

document.getElementById("err_prenume").innerHTML = "";
prenumeC = document.getElementById("prenume").value;
prenumeSablon = /^[A-Z][a-z]+$/;
if (!(prenumeC.match(prenumeSablon))) {
    document.getElementById("err_prenume").innerHTML = "Имя должно содержать
минимум 2 буквы - первая большая!!!";
    necazuri++;
}
```



ПРИМЕР ОПРЕДЕЛЕНИЯ И ИСПОЛЬЗОВАНИЯ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ. Продолжение

```
document.getElementById("err_tel").innerHTML ="";
    telefonC = document.getElementById("telefon").value;
    telefonSablon = /^+\d{11}$/
    if (!(telefonC.match(telefonSablon))) {
        document.getElementById("err_tel").innerHTML = "Номер телефона начинается
с '+' и потом следуют 11 цифр (+xxxxxxxxxxx)!!!";
        necazuri++;
    }

    return (necazuri==0);
}
</script>

</body>
</html>
```



ПРИМЕР ОПРЕДЕЛЕНИЯ И ИСПОЛЬЗОВАНИЯ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ. Стили

```
body{background-color: #404040;}
.err{color: orange; font-size: 12px;}
.corp{ width:600px; height:400px; }
label, legend, p{
    font-family:Arial, Helvetica, sans-serif;
    text-align:left; font-size:12px; color: #ffffcc; font-weight:bold; letter-spacing: 3px;}
.standard {
    font-family:Arial, Helvetica, sans-serif;
    text-align:left; font-size:10px; color: #ffffcc; }
input[type="submit"], input[type="reset"] {
    background-color: #ffffcc;
    border: 2px solid #ffffcc;
    color: #404040;
    padding: 10px 30px;
    font-weight: bold;
}
```



РЕЗУЛЬТАТ ПРИМЕРА

Контактные данные

Введите вашу фамилию:

Фамилия должна содержать минимум 2 буквы - первая большая!!!

Введите ваше имя:

Имя должно содержать минимум 2 буквы - первая большая!!!

Введите ваш номер телефона:

Номер телефона начинается с '+' и потом следуют 11 цифр (+xxxxxxxxxx)!!!

Обновить **Сохранить**



РЕЗУЛЬТАТ ПРИМЕРА с ошибками

Контактные данные

Введите вашу фамилию:

Фамилия должна содержать минимум 2 буквы - первая большая!!!

Фамилия должна содержать минимум 2 буквы - первая большая!!!

Введите ваше имя:

Имя должно содержать минимум 2 буквы - первая большая!!!

Имя должно содержать минимум 2 буквы - первая большая!!!

Введите ваш номер телефона:

Номер телефона начинается с '+' и потом следуют 11 цифр (+xxxxxxxxxxx)!!!

Номер телефона начинается с '+' и потом следуют 11 цифр (+xxxxxxxxxxx)!!!



РЕЗУЛЬТАТ ПРИМЕРА. Ошибки частично удалены

Контактные данные

Введите вашу фамилию:

Фамилия должна содержать минимум 2 буквы - первая большая!!!

Введите ваше имя:

Имя должно содержать минимум 2 буквы - первая большая!!!

Введите ваш номер телефона:

Номер телефона начинается с '+' и потом следуют 11 цифр
(+xxxxxxxxxxx)!!!

Номер телефона начинается с '+' и потом следуют 11 цифр
(+xxxxxxxxxxx)!!!



РЕЗУЛЬТАТ ПРИМЕРА

Контактные данные

Введите вашу фамилию:

Vasilev

Фамилия должна содержать минимум 2 буквы - первая большая!!!

Введите ваше имя:

Vasea

Имя должно содержать минимум 2 буквы - первая большая!!!

Введите ваш номер телефона:

+99988877766

Номер телефона начинается с '+' и потом следуют 11 цифр (+xxxxxxxxxx)!!!

Введите ваш адрес электронной почты:

vasilev.vasea@gmail.com

Введите ваш пароль:

1234567890

Введите ваш пароль повторно:

1234567890

Создать аккаунт

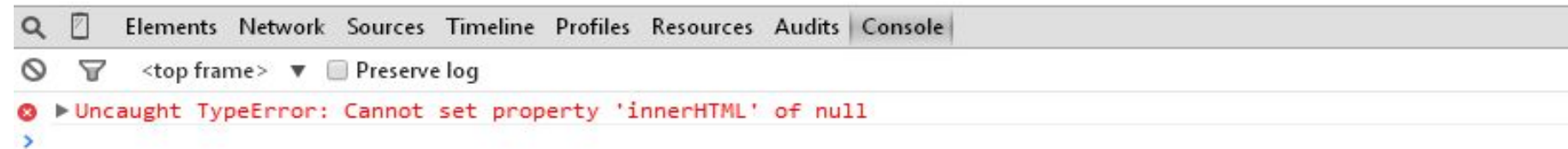
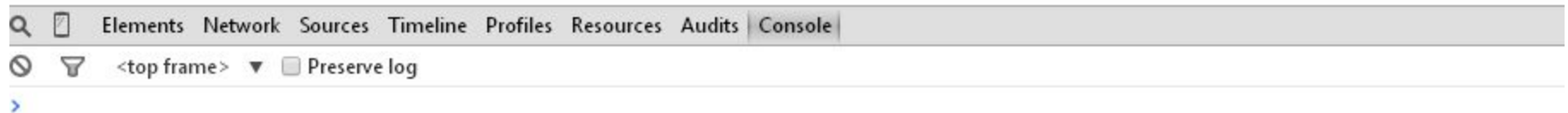
Уже есть аккаунт? Войти



JAVASCRIPT DEBUGGING

- В случае если появляются трудности в нахождении ошибок в скрипте используйте JS debugger
- Для активации debugger-а в браузере используйте клавишу быстрого доступа F12

Apasa pe buton pentru a vedea efectul ciclului **for..in**



JS-МЕТОДЫ ИСПОЛЬЗУЕМЫЕ ДЛЯ СОЗДАНИЯ ЭФФЕКТОВ

- http://www.w3schools.com/jquery/jquery_ref_effects.asp
- <http://api.jquery.com/category/effects/>
- <http://w.ict.nsc.ru/books/InetTechn/lab05/lab5-t.htm>

**УСПЕХОВ В СОЗДАНИИ ГРАФИЧЕСКИХ
ЭФФЕКТОВ!!!**



ДОПОЛНИТЕЛЬНЫЕ РЕКОММЕНДАЦИИ

- Для красивого и успешного создания front-end-a рекомендуется использовать:
 - Bootstrap
 - JSON
 - Backbone.js (<http://backbonejs.ru/>)





Знания

□ Что мы учили за последние 3 пары?

Способности/Навыки

Что мы можем сделать с полученными знаниями?

Отношения / Поведение

Что мы можем применить, как, где и для чего можно применить знания и навыки полученные за последние 3 урока?





-
- На следующей неделе, 06.05.2015, пишем 2-ю аттестацию
 - Работа будет состоять из задач решение которых демонстрирует ваши знания, навыки и отношения из области CSS и JavaScript
 - Не забудьте повторить темы 7-12, в том числе примеры с лабораторных работ!!!

