Державний університет
«Житомирська політехніка»

Лекція 3.1.
Регулярні вирази

Регулярні вирази - мова для пошуку та маніпуляції над підрядками у тексті.

/z[aoue]r/

zar, zor, zur, zer

але не zir, dpo, aga

/^([a-z0-9_\.-]+)@([a-z0-9_\.-]+)\.([az\.]{2,6})$/

****	**перетворює спеціальний символ на звичайний і навпаки**
.	будь-який символ, окрім перекладу рядка
*****	повторення попереднього символу 0 і більше разів
+	повторення попереднього символу 1 і більше разів
?	повторення попереднього символу 0 або 1 раз
\d	будь-яка цифра
\w	будь-який словесний символ (літери, цифри та _)
[XYZ]	будь-який символ із зазначених
[XYZ]+	один або більше символів із зазначених
$	кінець даних
^	початок даних
[^az]	НЕ мала літера (всередині класу ^ означає НЕ)
( )	дужки, що запам'ятовують
|	або
{m, n}	від m до n повторень попереднього символу

В Javascript є конструктор **RegExp . new RegExp (“ w + c ”).**

```
new RegExp("pattern"[, флаги])
/pattern/прапори
```

```
var reg = new RegExp("ab+c", "i")
var reg = /ab+c/i
```

прапори:

"i" - ігнорувати регістр символів

“g” – глобальний пошук

"m" - багаторядковий пошук

Об'єкт	Метод	Опис	Повертає
RegExp	test	Чи є збіг у рядку	true / false
RegExp	exec	Пошук збігів у рядку	масив
String	search	Пошук збігів у рядку	індекс збігу /-1
String	match	Пошук збігів у рядку	масив / null
String	replace	Пошук збігів та заміна	рядок
String	split	Розбиття рядка на масив підрядок	масив

```
var reg = /^([a-z0-9_\.-]+)@([a-z0-9_\.-]+)\.([a-z\.]{2,6})$/
```

Графічний онлайн візуалізатор виразів - **http://www.regexper.com**

Метод setTimeout

```
var timerId = setTimeout(func/code, delay[, arg1, arg2...])
```

Параметри:
func / code

Delay – Затримка в мілісекундах, 1000 мілісекунд дорівнюють 1 секунді.

arg 1, arg 2… - аргументи, які необхідно передати функції. Не підтримуються у браузерах IE нижче 9 версії.

```
function func() {
   alert('Привіт');
}
setTimeout(func, 1000);
```

```
setTimeout("alert('Привіт')", 1000)
```

Метод setInterval

```
var timerId = setInterval(func/code, delay[, arg1, arg2...])
```

```
<input type="button" onClick="clearInterval(timer)" value="Стоп">
  <script>
    var i = 1;
    var timer = setInterval(function() {alert(i++)}, 2000);
  </script>
```

1. Масиви

```
var arr = [];
```
- порожній масив

```
var groups = ["ПІ-51", "ПІ-52","КІ-1"];
```

```
alert( groups[0] );
```

```
groups[3] = "ПІ-47в";
```

```
alert(groups.length); // 4
```

Масив може містити значення будь-яких типів:

```
var arr = [true, 1.5, "string", [1,2,3]];
```

Державний університет
«Житомирська політехніка»

Лекція 3.2.
Масиви. Об'єкти

1. Масиви

```
var arr = [];
```
- порожній масив

```
var groups = ["ПІ-51", "ПІ-52","КІ-1"];
```

```
alert( groups[0] );
```

```
groups[3] = "ПІ-47в";
```

```
alert(groups.length); // 4
```

Масив може містити значення будь-яких типів:

```
var arr = [true, 1.5, "string", [1,2,3]];
```

Масиви можуть використовуватися як черги та стеки.

Додавання та отримання з правої сторони:

```
var cities = ["Житомир", "Київ", "Вінниця"];
console.log(cities.pop()); // Вінниця
```

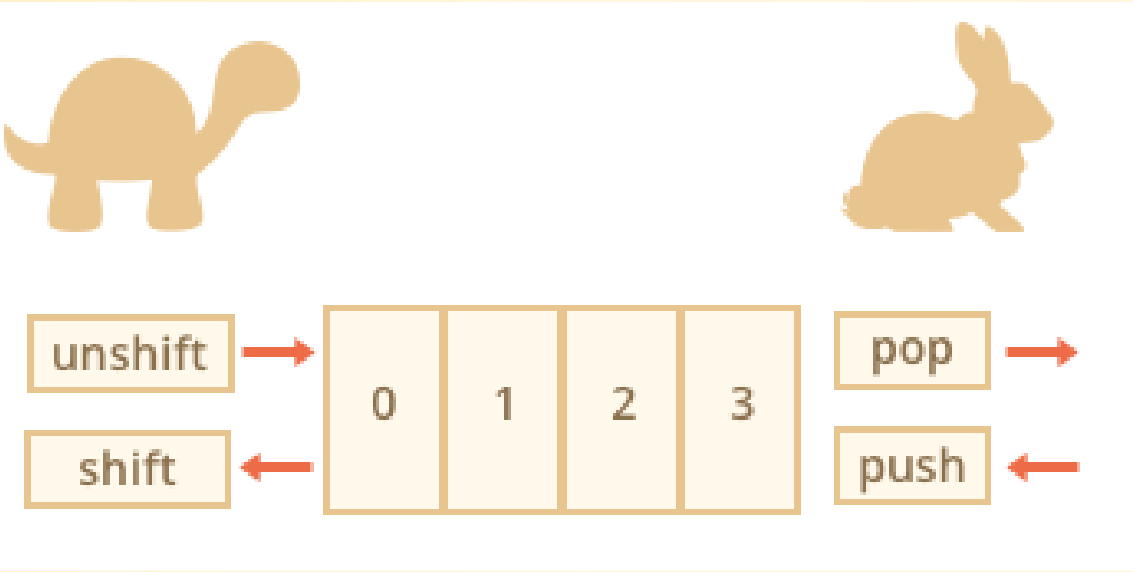
```
cities.push("Хмельницький");
```

```
console.log( cities ); // ["Житомир", "Київ",
                       //  "Хмельницький"]
```

Додавання та отримання з лівої сторони:

```
var cities = ["Житомир", "Київ", "Вінниця"];
console.log( cities.shift() ); // Житомир
cities.unshift("Хмельницький");
console.log( cities ); // ["Хмельницький",
                       //  "Київ", "Вінниця"]
```

push/pop виконуються швидко,
unshift/shift виконуються повільно



Метод shift:

1. Видаляє нульовий елемент
2. Зміщує всі елементи вліво
3. Оновлює властивість length

У масиві можна пропускати елементи:

```
var arr = [];
arr[0] = 0;
arr[5] = 5;
console.log( arr );
```

Довжина масиву length – це не кількість елементів масиву, а останній індекс + 1

```
var arr = [];
arr[0] = 0;
arr[5] = 5;
console.log( arr ); // 6
```

Властивість length доступна для запису:

```
var arr = [1, 2, 3, 4, 5];

arr.length = 2; // залишити 2 елементи
alert( arr ); // [1, 2]

arr.length = 5;
alert( arr[3] ); // undefined
```

```
arr.length = 0; // очистити масив
```

Масив можна створювати через конструктор:

```
var cities = new Array("Житомир", "Київ", "Вінниця");
```

Це аналогічно до

```
var cities = ["Житомир", "Київ", "Вінниця"];
```

Якщо в конструктор передати одне ціле число, то буде встановлено довжину масиву:

```
var groups = new Array(5);
console.log(groups.length);
console.log(groups);
```

Двовимірні масиви:

```
var matr =
        [
                [1, 2, 3, 4],
                [5, 6, 7],
                [8, 9, "ten", 11.5]
        ];
var matr = new Array(
        new Array(1, 2, 3, 4),
        new Array(5, 6, 7),
        new Array(8, 9, "ten", 11.5)
);
matr[1][2] = 17; // замість 7 пишемо 17
console.log(matr[2][2]); // "ten"
```

Звернення до неіснуючого елемента

```
var matr = [
                [1, 2, 3, 4],
                [5, 6, 7],
                [8, 9, "ten", 11.5]
           ];
console.log(matr[2][5]); // undefined
```

```
matr[2]     // [8, 9, "ten", 11.5]
```

```
matr[2][5] // undefined
```

```
console.log(matr[3][2]); // Error!!!
```

```
matr[3]     // undefined
```

```
matr[3][i] // до undefined неможна
           // застосувати індексування
```

Методи масивів:

arr.join('роздільник') – об'єднує масив у рядок;

arr.splice(index[, deleteCount, elem1, ..., elemN]) – видаляє **deleteCount** елементів починаючи з індексу index, а потім на їх місце додає елементи **elem1**, …, **elemN**

```
var arr = ["Я", "вивчаю", "JavaScript"];
arr.splice(1, 1);
// видалити 1 елемент починаючи з індексу 1
console.log( arr ); //  ["Я", "JavaScript"]
```

```
var arr = ["Я", "вивчаю", "JavaScript"];
arr.splice(0, 2, "Ми", "вивчаємо")
console.log( arr );
//  ["Ми", "вивчаємо", "JavaScript"]
```

```
var arr =["Я", "зараз", "вивчаю", "JavaScript"];
var removed = arr.splice(2, 2);
console.log( removed ); //  ["вивчаю", "JavaScript"]
```

```
var arr = ["Я", "вивчаю", "JavaScript"];
arr.splice(2, 0, "складну", "мову");
console.log( arr ); //  ["Я", "вивчаю",
          // "складну", "мову", "JavaScript"]
```

Від’ємний індекс відраховує елементи з правої сторони:

```
var arr = [1, 2, 5];
arr.splice(-1, 0, 3, 4);
alert( arr ); // результат: 1,2,3,4,5
```

`arr.slice(begin, end)` – копіює частину масиву

`arr.sort()` – сортування масиву

```
var arr = [ 1, 2, 15 ];
arr.sort();
alert( arr );  // 1, 15, 2
```

Для порівняння цілих чисел треба використовувати компаратор:

```
function compareNumeric(a, b) {
    if (a > b) return 1;
    if (a < b) return -1;
}
var arr = [ 1, 2, 15 ];
arr.sort(compareNumeric);
console.log(arr);  // 1, 2, 15
```

arr.reverse() – зміна порядку слідування елементів на протилежну;

arr.concat(value1, value2, … valueN) – додавання елементів у кінець масиву;

```
var arr = [1, 2];
var newArr = arr.concat([3, 4], 5);
// arr.concat(3,4,5)
alert( newArr ); // 1,2,3,4,5
```

arr.indexOf(зн), arr.lastIndexOf(зн) – пошук значень у масиві

```
var arr = [1, 0, false];
alert( arr.indexOf(0) );       // 1
alert( arr.indexOf(false) );   // 2
alert( arr.indexOf(null) );    // -1
```

Внутрішня реалізація indexOf/lastIndexOf здійснює повний перебір, аналогічний до циклу for по масиву