Інтернет програмування

лек 2

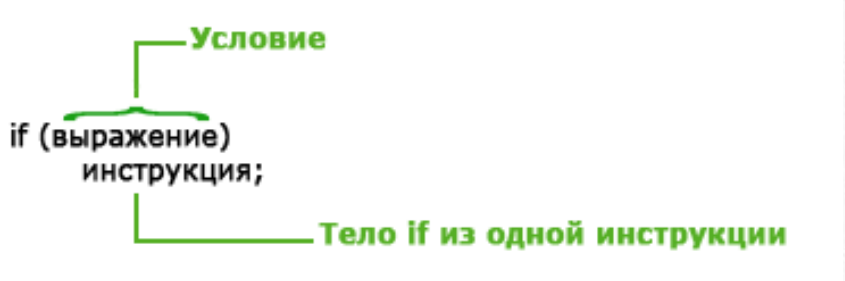
Умовні оператори, функції, цикли

Зав.каф.КН, PhD Граф М.С.

План курсу

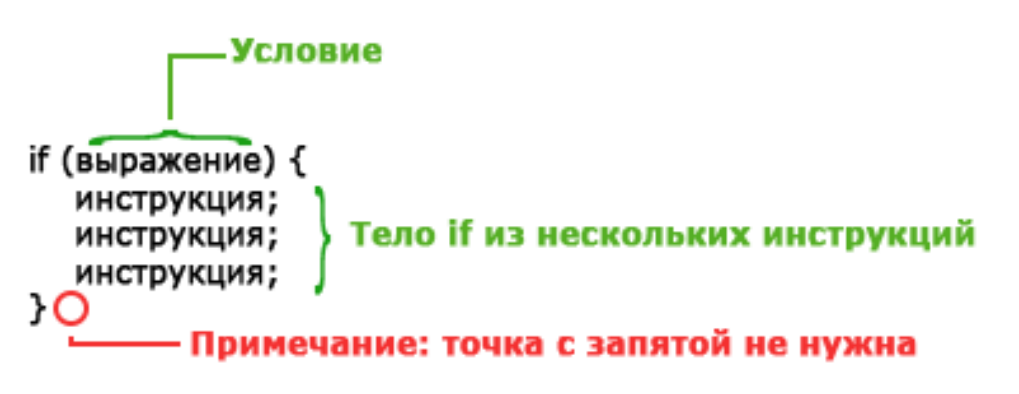
1. **Інструкція if**
2. **Функції**

Інструкція if має дві форми. Перша:




```
var year = prompt('В кому році Бандера вступив до університету?', '');
if (year != 2022) alert( 'Невірно!' );
```

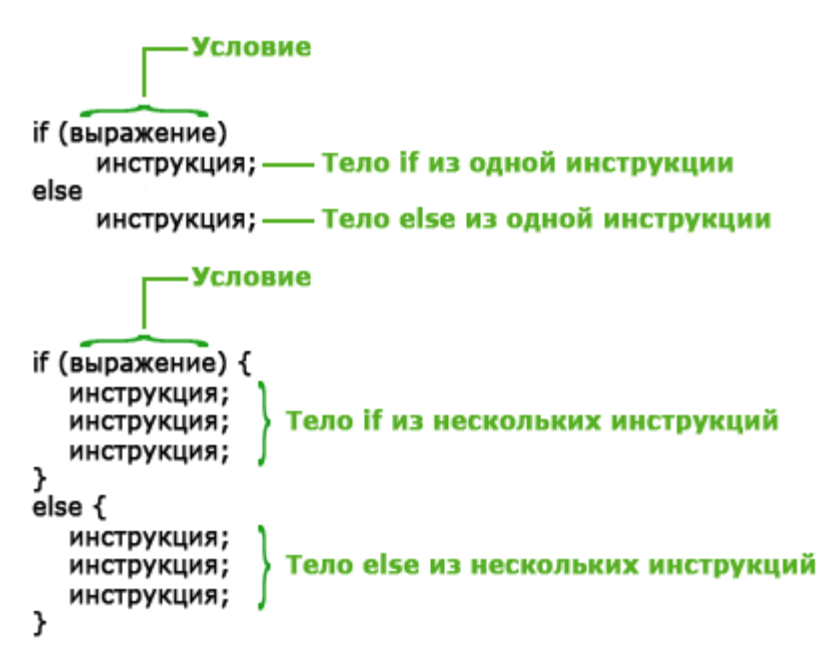
Синтаксис JavaScript дозволяє вставити тільки одну інструкцію, що виконується після інструкції if, проте якщо потрібно виконати більше однієї інструкції, її завжди можна замінити блоком інструкцій:




```
if (year != 2011) {
    alert( 'А вот..' );
    alert( '..і невірно!' );
}
```

У логічному контексті:
•Число 0, порожній рядок "", null і undefined, а також NaN false,
•Інші значення - true.

З інструкцією if може використовуватися ключове слово else, яке дозволяє додати інструкцію (або блок коду), яка виконується, якщо умова має хибне значення. Друга форма інструкції if:




```
var num = 15;
if (num > 10)
    alert("число " + num + " більше 10");
else
    alert("число " + num + " менше 10");
```

Часто практично буває необхідно перевірити кілька варіантів умов. Для цього можна застосувати таку конструкцію:

```
var num = 2;
if (num == 1) {
    alert("значення num: " + num);
} else if (num == 2) {
    alert("значення num: " + num);
} else if (num == 3) {
    alert("значення num: " + num);
} else {
    alert("Не знаю такого числа!");
}
```

Умовний оператор

Оператор	Операція	Типи значень
?:	Вибір другого чи третього операнда	булеве, будь-яке, будь-яке → будь-яке

умова? значення1: значення2

Умовний оператор - це єдиний тернарний оператор JavaScript. Перший операнд передує символу ?, Другий - між ? і:, третій - після:.

Якщо перший операнд повертає справжнє значення, обчислюється і повертається значення другого операнда. Якщо перший операнд має хибне значення, то обчислюється та повертається значення третього операнда.

```
alert (true ? 5 : 10);
```

Умовний оператор часто використовується як коротший варіант інструкції if/else.

```
var a, b, num = 2;
if (num)
    a = 5;
else
    a = 10;
```

те саме, з умовним оператором:

```
var b = num ? 5 : 10;
alert("a: " + a + "<br>");
alert("b: " + b);
```

Функції

Функція визначається за допомогою ключового слова **function** , за яким вказуються такі компоненти:

•Ідентифікатор, який визначає назву функції.
•Блок коду укладений у пару фігурних дужок. Він є тілом функції та виконується при кожному виклику функції.
•Пара круглих дужок, для параметрів функції, що розділяються комами.

function ім’я_функції(параметри) { інструкції }

Параметрів у функції може бути вказана будь-яка кількість:

```
function test(a,b,c) {...} // a,b,c - параметри функції
```

При виклику функції їй можуть передаватися значення, якими будуть ініціалізовані параметри. Значення, що передаються під час виклику функції, називаються аргументами.

```
function sum(a,b) {
    var c = a + b;
    alert(c);
}
sum(5, 7);
```

Якщо кількість аргументів відрізняється від числа параметрів, жодної помилки не відбувається

```
function test(a,b,c) { ... }
```

Інструкція return

Інструкція return всередині функції служить визначення значення, повертаного функцією.

```
return выражение;
```

Для подальшого використання значення, що повертається, результат виконання функції можна присвоїти наприклад змінної:

```
function calc(a) {
    return a * a;
}
var x = calc(5);
document.write(x);
```

Інструкція return може бути розташована будь-де функції. Як тільки буде досягнуто інструкції return, функція повертає значення і негайно завершує своє виконання. Код, розташований у тілі функції після інструкції return, буде проігноровано.

```
let a = 1;
function test() {
    ++a;
    return;
    ++a;
}
test();
alert(a);  // => 2
```

Всередині функції може бути використано кілька інструкцій return, наприклад, якщо значення, що повертається, залежить від деяких умов виконання:

```
function check(a, b) {
    if(a > b) return a;
    else return b;
}
document.write(check(3, 5));
```

Якщо інструкція return не вказана або не вказано значення, що повертається, то функція поверне значення undefined.

Інструкція return не вказана

```
function bar() { alert("функція bar виконалась"); }
```

Значення, що повертається, не вказано

```
function test(a) {
    if(!a) return;
    alert(a);
}
var a = bar(); // undefined
var b = test(); // undefined
alert("a: " + a + ",   b: " + b);
```

Визначення функції неявно "піднімається" на початок сценарію або вміщає її (зовнішньої) функції, завдяки чому функцію можна викликати ще до того, як вона буде визначена.

```
test();
bar();
function test() {
    alert("Hello!");
}
function bar() {
    var str = "Hello again!";
    test();
    function test() {
        alert(str);
    }
}
```

Таким чином, функції вище еквівалентні визначенням функцій наведеним нижче, в яких визначення "підняті" на початок.

```
function test() {
    alert("Hello!");
}
function bar() {
    function test() {
        document.write(str);
    }
    var str = "Hello again!";
    test();
}
test();
bar();
```

Функція isNaN()

Таким чином, функції вище еквівалентні визначенням функцій наведеним нижче, в яких визначення "підняті" на початок.
Вона повертає true, якщо аргумент має значення NaN або якщо є нечисловим значенням, таким як рядок або об'єкт.
Синтаксис:

```
isNaN(testValue);
alert(isNaN("123"));           // false
alert(isNaN(-1.23));           // false
alert(isNaN("5"-2));           // false
alert(isNaN(0));               // false
alert(isNaN("Hello"))
```

Метод parseFloat

Аналізує рядковий аргумент та повертає число з плаваючою точкою.

```
parseFloat(string)
```

```
parseFloat("3.14"); // 3.14
parseFloat("тест") // NaN
```

Метод parseInt

Аналізує рядковий аргумент та повертає ціле число, визначене як основа.

```
parseInt(string [,radix])
```

Якщо radix не вказано або дорівнює 0, то JavaScript передбачає наступне:

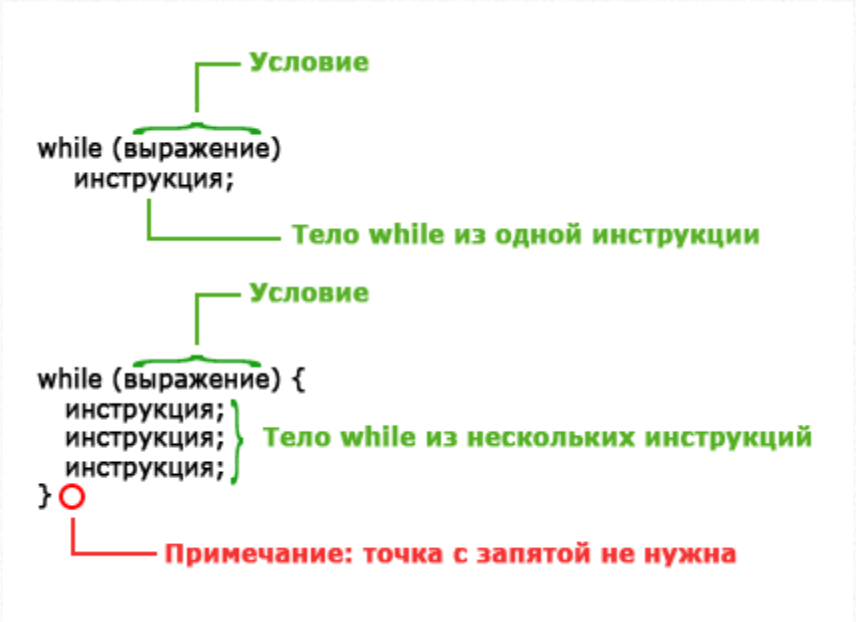
- Якщо вхідний рядок починається з "0x", то radix = 16
- Якщо вхідний рядок починається з "0", radix = 8. Цей пункт залежить від реалізації і в деяких браузерах (Google Chrome) відсутній.
- У будь-якому іншому випадку radix=10.

```
parseInt(" 0xF", 16)
parseInt(" F", 16)
parseInt("17", 8)
parseInt(021, 8)
parseInt("015", 10)
parseInt(15.99, 10)
parseInt("FXX123", 16)
parseInt("1111", 2)
```

Цикл

Цикл- це інструкція, що дозволяє повторювати виконання деяких дій (інструкцій) певну кількість разів. Кожне окреме виконання інструкцій у тілі циклу називається ітерація

Цикл while

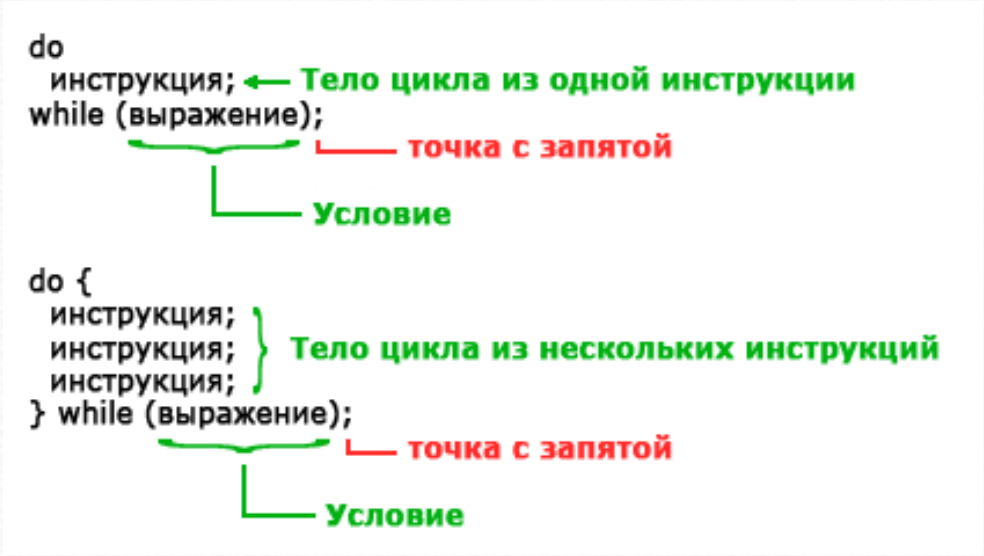

Условие
while (выражение)
инструкция;
Тело while из одной инструкции
Условие
while (выражение) {
инструкция;
инструкция;
инструкция;
}
Тело while из нескольких инструкций
Примечание: точка с запятой не нужна

Приклад циклу while:

```
let count = 0; // визначаємо змінну рахівник
while (count < 3) {
    document.write(count + " ");
    count++;     // якщо з коду прибрати даний рядок, то цикл буде нескінченним
}
```

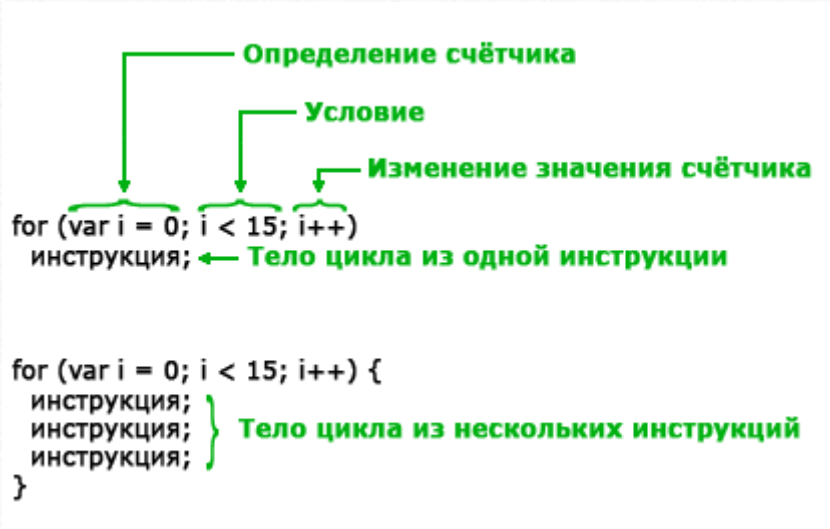
```
let i = 3;
while (i) { // при і, рівному 0, значення в скобках буде false і цикл зупиняється
    alert( i );
    i--;
}
```

Цикл do/while




```
let count = 0;
do {
    count++;
    alert(count);
} while(count < 5);
```

Цикл for



Приклад, у якому виводяться цифри від 0 до 3:

```
for (let count = 0; count < 4; count++)
    alert(count + " ");
```

```
let i = 0;
for (; i < 4; i++) ...

let i = 0;
for (; i < 4; ) ...

for (let i = 1; /* нема умови */ ; i++) ...
```

Це еквівалентно наступному коду

```
for (let i = 1; true; i++) ...
```

```
for (i = 1; i < 4, false; i++) ...
for (let i = 1, j = 5; i <= 5; i++, j--)
    document.write(i + " " + j +"<br>");
```

Вложенные циклы

```
for(let i = 0; i < 3; i++) {
    alert("Частина зовнішнього циклу. <br>");
    for(let j = 0; j < 2; j++) {
        alert("Частина вкладеного циклу. <br>");
    }
}
let i = j = 0;
while (i < 3) {
    j = 0;
    document.write(" Частина зовнішнього циклу. <br>");
    while(j < 1) {
        document.write(" Частина вкладеного циклу <br>");
        j++;
    }
    i++;
}
```

Інструкції break і continue

```
for(let i = -10; i <= 10; i++) {
    if(i > 0) break;
    alert(i + " ");
}
alert("Готово!");
var j = -1;
for(var i = 0; i < 3; i++) {
    document.write("Частина зовнішнього циклу. <br>");
    while (j < 5) {
        j++;
        if (j == 2 || j == 3) break;
        document.write("j: "+ j +"<br>");
    }
}
```

Коли інструкція виконується **continue** , поточна ітерація циклу переривається і починається наступна. Для різних циклів інструкція дає різний ефект: Інструкція **continue** схожа на інструкцію **break** , проте замість виходу з циклу вона запускає нову ітерацію циклу. Інструкція continue може використовуватися лише у циклах. Синтаксис інструкції continue також простий, як і синтаксис інструкції break:

- Після виконання інструкції **continue** у циклі **while** перевіряється умова виконання, і якщо вона має значення true, тіло циклу виконується.
- У циклі for після виконання інструкції continue **спочатку** обчислюється третій вираз, і потім відбувається перевірка умови.
- У циклі do/while після виконання інструкції **continue** відбувається перехід у кінець циклу та перевіряється умова виконання, якщо воно має значення true, тіло циклу виконується.

```
for (let i = 0; i <= 10; i++) {
    if((i % 2) != 0) continue;
    alert(i);
}
```

Інструкція switch у JavaScript

```
switch (вираз) {
    case константа: інструкції;
    ...
    case константа: інструкції;
    default: інструкції;
}
```

Наявність блоку default не обов'язково, якщо його немає і не знайдено жодної відповідності, то жодних інструкцій не буде виконано:

```
let x = 1;
switch (x) {
    case 1:
        alert("x рівне 1");
        break;
    case 2:
        alert("x рівне 2");
        break;
    case 3:
        alert("x рівне 3");
        break;
    default:
        alert("x > 3");
}
```

Приклад без інструкції break:

```
let x = 2;
switch (x) {
    case 1:
        document.write("x рівен 1");
    case 2:
        document.write("x рівен 2");
    case 3:
        document.write("x рівен 3");
    default:
        document.write("x > 3");
}
```

```
let x = 3;
switch (x) {
    case 1:
    case 2:
    case 3:
        document.write("x рівне 1, 2 або 3");
        break;
    case 7:
        document.write("x рівне 7");
        break;
}
```

```
let x = 3;
switch (x) {
    case 1: case 2: case 3:
        document.write("x рівне 1, 2 або 3");
        break;
    case 7:
        document.write("x рівне 7");
        break;
}
```

Об'єкт Math

Синтаксис:

Math.константа

Math.функція()

Тригонометричні методи	
cos(x)	косинус числа x
sin(x)	синус числа x
tan(x)	тангенс числа x
acos(x)	арккосинус числа x
asin(x)	арксинус числа x
atan(x)	арктангенс числа x
Методи обчислень	
exp(x)	e^x
log(x)	натуральний логарифм ln(x)
pow(x, y)	x^y
sqrt(x)	квадратний корінь від x
Константи	
E	Число e – основа натурального логарифму (≈ 2.72)
LN10	Число ln(10) ≈ 2.3
LN2	Число ln(2) ≈ 0.69
LOG10E	Число lg(e) ≈ 0.43
LOG2E	Число $\log_2(e)$ ≈ 1.44
PI	Число π ≈ 3.14
SQRT1_2	Квадратний корінь від ½ ≈ 0.71

Дякую за увагу!