Biến, kiểu dữ liệu và toán tử

Khóa học: NỀN TẢNG LẬP TRÌNH

Mục tiêu

- Trình bày được ý nghĩa của ngôn ngữ lập trình
- Trình bày được cách thực thi của JavaScript
- Trình bày được cú pháp của JavaScript
- Sử dụng được các hàm nhập và xuất của JavaScript
- Sử dụng được chú thích trong JavaScript
- Tạo được ứng dụng JavaScript đầu tiên
- Trình bày được khái niệm biến
- Trình bày được cú pháp khai báo biến
- Trình bày được khái niệm kiểu dữ liệu
- Trình bày được các toán tử thông dụng
- Khai báo và sử dụng được biến
- Sử dụng được các kiểu dữ liệu
- Sử dụng được các toán tử cơ bản

JavaScript

- JavaScript là một ngôn ngữ lập trình được sử dụng nhiều trên các website
- JavaScript chạy trên trình duyệt (nếu trình duyệt không hỗ trợ thì JavaScript không chạy được)
- *Ngày nay, JavaScript còn được sử dụng để tạo ra các ứng dụng ở phía back-end*
- Sử dụng Javascript trên trang web, chúng ta có thể:
  - Thay đổi nội dung trang web
  - Thay đổi giao diện trang web
  - Tăng tính tương tác của trang web

Chương trình máy tính

- Chương trình máy tính là các chuỗi **hướng dẫn** (instruction) để máy tính **thực thi** (execute)
- Trong một ngôn ngữ lập trình, các chuỗi hướng dẫn được gọi là **câu lệnh** (statement)

Câu lệnh (statement)

- Câu lệnh được tạo nên từ các thành phần:
  - Giá trị (value)
  - Toán tử (operator)
  - Biểu thức (expression)
  - Từ khóa (keyword)
  - Chú thích (comment)
- Ví dụ, đây là 4 câu lệnh trong JavaScript:

```
let x, y, z;
x = 5;
y = 6;
z = x + y;
```

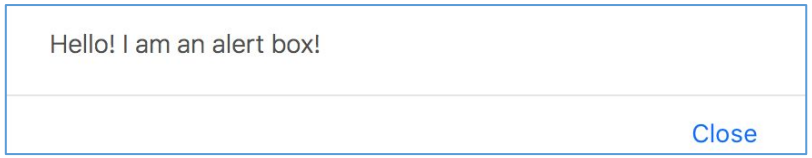
Chú thích (comment)

- Chú thích được dùng để giải thích ý nghĩa của một đoạn mã nào đó
- Chú thích không được thực thi (execute)
- Cách viết chú thích trên một dòng sử dụng `//`
- Cách viết chú thích trên nhiều dòng sử dụng `/*...*/`

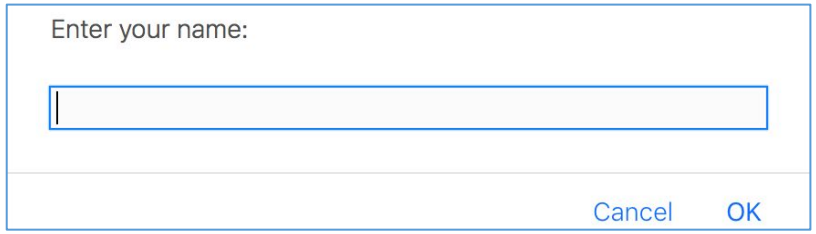
- ***Lưu ý:*** *Thêm chú thích vào trong mã nguồn để giải thích cho mã nguồn dễ hiểu hơn là một việc tốt. Tuy nhiên, sẽ tốt hơn nếu mã nguồn của chúng ta không cần có chú thích mà vẫn dễ hiểu.*

Các hàm có sẵn: alert(), prompt(), confirm()

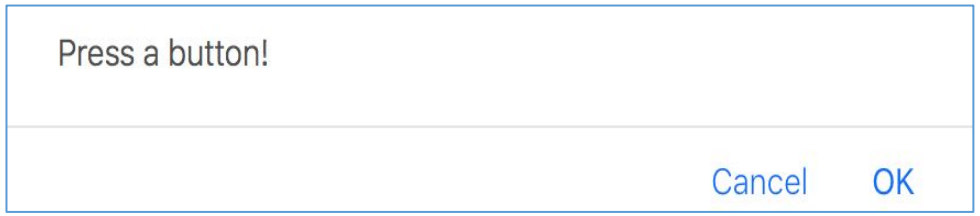
```
<!DOCTYPE html>
<html>
<body>

<script>
alert("Hello! I am an alert box!");
</script>

</body>
</html>
```




```
<!DOCTYPE html>
<html>
<body>

<script>
prompt("Enter your name: ");
</script>

</body>
</html>
```




```
<!DOCTYPE html>
<html>
<body>

<script>

confirm("Press a button!");

</script>

</body>
</html>
```



Hiển thị kết quả thực thi

- Một số cách để hiển thị kết quả thực thi trong Javascript:
  - Sử dụng hàm **document.write()**
  - Sử dụng hàm **alert()**
  - Sử dụng hàm **console.log()**

Ngoài ra, có thể sử dụng HTML DOM (Nội dung này sẽ được hướng dẫn chi tiết trong bài học tiếp theo)

- *Thuộc tính **innerHTML, innerText***

Demo

Tạo chương trình JavaScript đầu tiên
Sử dụng các hàm nhập dữ liệu
Sử dụng các hàm xuất dữ liệu

Thảo luận

Biến và Hằng

Định danh

Biến

Hằng

Quy ước đặt tên

Biến (variable)

- **Biến** là một tên gọi được gắn cho một vùng nhớ chứa dữ liệu
- Dữ liệu được lưu trữ trong vùng nhớ của biến được gọi là **giá trị** (value)
- Có thể truy nhập, gán hay thay đổi giá trị của biến
- Khi gán một giá trị mới thì giá trị cũ sẽ bị ghi đè lên
- Cần phải khai báo biến trước khi sử dụng

Ví dụ về biến

```
BEGIN
    DISPLAY “Enter 2 numbers: ”
    INPUT A, B
    C = A + B
    DISPLAY C
END
```

- A, B và C là các biến trong đoạn mã giả trên
- Tên biến giúp chúng ta truy cập vào bộ nhớ mà không cần dùng địa chỉ của chúng
- Hệ điều hành đảm nhiệm việc cấp bộ nhớ còn trống cho những biến này
- Để tham chiếu đến một giá trị cụ thể trong bộ nhớ, chúng ta chỉ cần dùng tên của biến

Khai báo và gán giá trị cho biến

- Từ khoá `let` được dùng để **khai báo** biến
  (Ngoài ra, có thể sử dụng từ khóa `var`)
- x là tên biến
- Dấu bằng (=) được dùng để **gán giá trị** cho biến

Cú pháp:

let variableName;

variableName = value;

Ví dụ:

```
let x;

y = 6;
```

Đặt tên cho biến

- Tên biến phải bắt đầu bằng một ký tự alphabet (a-zA-z_)
- Theo sau ký tự đầu có thể là các ký tự chữ, số ...
- Nên tránh đặt tên biến trùng tên các từ khoá
- Tên biến nên mô tả được ý nghĩa của nó
- Tránh dùng các ký tự gây lầm lẫn
- Tên biến có phân biệt chữ hoa và chữ thường
- Nên áp dụng các quy ước đặt tên biến chuẩn khi lập trình

Đặt tên cho biến: Ví dụ

- Ví dụ đặt tên biến đúng
  - arena
  - s_count
  - marks40
  - class_one
- Ví dụ đặt tên biến sai
  - 1sttest
  - oh!god
  - start... end

Giá trị của biến

- Ví dụ
  - 5 **số / giá trị số nguyên** (integer)
  - 5.3 **số / giá trị số thập phân** (decimal)
  - “Black” **Giá trị chuỗi** (string)
  - ‘C’ **Giá trị ký tự** (character)
  - true và false là các **giá trị logic** (boolean)

Hằng (constant)

- Hằng là một tên gọi đại diện cho một giá trị cố định
- Giá trị của hằng không thể thay đổi
- Giá trị của hằng cần phải được gán tại thời điểm khai báo
- Ví dụ, sử dụng hằng PI thay cho giá trị 3.14159:

```
let area = radius * radius * 3.14159;
```

Được thay bằng:

```
const PI = 3.14159;
let area = radius * radius * PI;
```

Khai báo hằng

- Cú pháp khai báo hằng:

```
const CONSTANT_NAME = value;
```

Trong đó:

- *const* là từ khoá bắt buộc để khai báo hằng
- *CONSTANT_NAME* là tên của hằng
- *value* là giá trị của hằng

Demo

Biến, kiểu dữ liệu
Các kiểu dữ liệu nguyên thuỷ
Chuỗi
Giá trị mặc định

Kiểu dữ liệu (Data Type)

- Kiểu dữ liệu là một cách phân loại dữ liệu cho trình biên dịch hoặc thông dịch hiểu các lập trình viên muốn sử dụng dữ liệu.
- Kiểu dữ liệu mô tả loại dữ liệu sẽ được lưu trong biến
- Các kiểu dữ liệu khác nhau được lưu trữ trong biến là:
  - Số (number)
    - Số nguyên: 10 hay 83839
    - Số thực: 15.33 hay 23.6677
    - Số dương: 3, 4
    - Số âm: -6, -7
  - Chuỗi (string): "Hello"
  - Ký tự: 'A'
  - Logic (boolean): true, false

Kiểu dữ liệu (Data Type)

- Một kiểu dữ liệu cung cấp một bộ các giá trị mà từ đó một biểu thức (như biến, hàm ...) có thể lấy giá trị của nó
- Trong JavaScript khi khai báo biến và gán cho biến một giá trị đồng nghĩa xác định kiểu dữ liệu cho biến đó.
- Ví dụ:
  - var x = 10; // x mang giá trị 10, kiểu dữ liệu của x là số nguyên
  - var gender = true; // gender mang giá trị true, kiểu dữ liệu của gender là kiểu boolean

Chuỗi

- Chuỗi bao gồm các ký tự liên tiếp nhau
- Có thể khai báo chuỗi sử dụng dấu nháy đơn hoặc nháy kép
- Ví dụ:

```
let answer = "It's alright";
let answer = "He is called 'Johnny'";
let answer = 'He is called "Johnny"';
```

Số

- Có thể sử dụng số nguyên hoặc số thập phân
- Ví dụ:

```
let x1 = 34.00;
let x2 = 34;
```

Boolean

- Kiểu dữ liệu boolean chỉ có hai giá trị là **true** và **false**
- Tất cả mọi thứ có giá trị đều là **true**
- Tất cả mọi thứ không có giá trị đều là **false**

true

```
100

3.14

-15

"Hello"

"false"

7 + 1 + 3.14
```

false

```
let x = false;
Boolean(x);  // return false

let x = 0;
Boolean(x);  // return false

let x = "";
Boolean(x);  // return false

let x;
Boolean(x);  // return false

let x = null;
Boolean(x);  // return false
```

Thảo luận

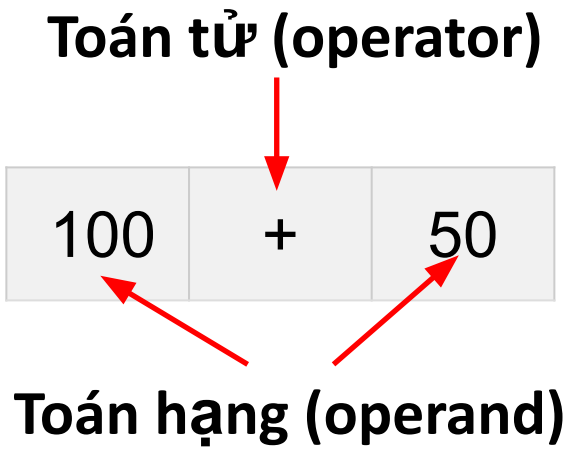
Toán tử gán
Toán tử số học
Toán tử so sánh
Toán tử logic

Toán tử (Operator)

- Toán tử là các ký hiệu được sử dụng để thực hiện các thao tác trong các biểu thức và sinh ra kết quả cuối
- Có nhiều loại toán tử khác nhau:
  - Toán tử toán học
  - Toán tử gán
  - Toán tử cộng chuỗi
  - Toán tử so sánh
  - Toán tử logic
  - Toán tử `typeof`

Toán tử toán học (arithmetic)

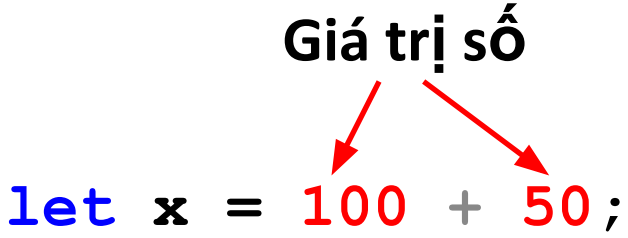
- Toán tử toán học được sử dụng trong các biểu thức toán học
- Toán tử toán học được sử dụng trên các giá trị số (hoặc là các biến kiểu số)
- Toán tử toán học thông thường có 2 toán hạng



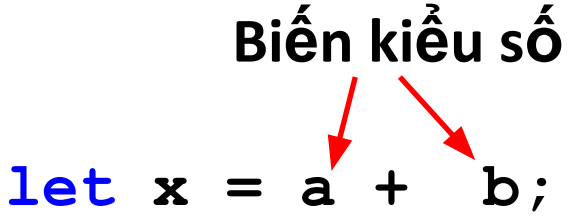
Toán tử	Mô tả
+	Cộng
-	Trừ
*	Nhân
/	Chia
%	Chia lấy phần dư (Modulus)
++	Tăng 1 giá trị
--	Giảm 1 giá trị

Toán tử toán học: Ví dụ

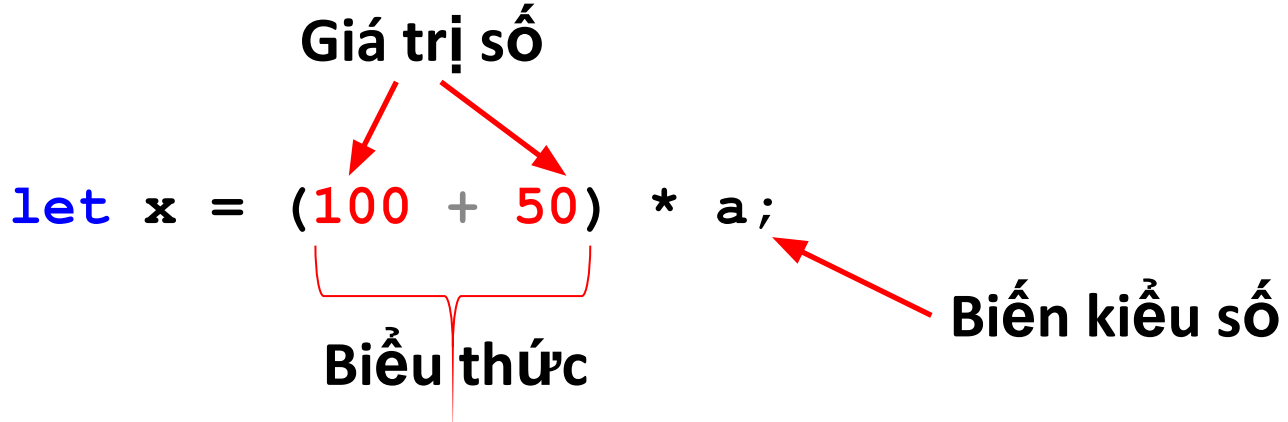
- Sử dụng với các giá trị:



- Sử dụng với các biến:

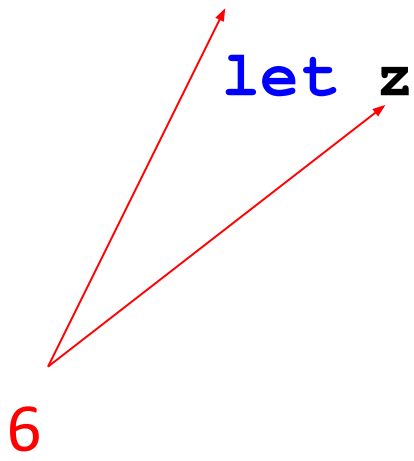


- Sử dụng với biểu thức:



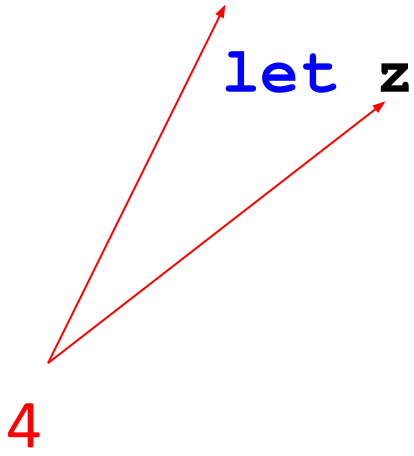
Toán tử tăng: Ví dụ

```
let x = 5;

x++;

let z = x;
```



Toán tử giảm: Ví dụ

```
let x = 5;

x--;

let z = x;
```



Toán tử gán (assignment)

- Toán tử gán được sử dụng để gán giá trị cho một biến
- Toán tử gán có thể sử dụng với tất cả các kiểu dữ liệu
- Ví dụ:

```
let x = 10;
```

Operator	Example	Same As
=	x = y	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

Toán tử phần trăm bằng: ví dụ

```
let x = 10;
x %= 5;   //Tương đương với x = x % 5
```


0

Toán tử cộng chuỗi (string concatenate)

- Toán tử cộng chuỗi được sử dụng để nối hai chuỗi
- Có thể nối chuỗi với số

```
txt1 = "John";
txt2 = "Doe";
txt3 = txt1 + " " + txt2;
```

"John Doe"

```
x = 5 + 5;
y = "5" + 5;
z = "Hello" + 5;
```

10

"55"

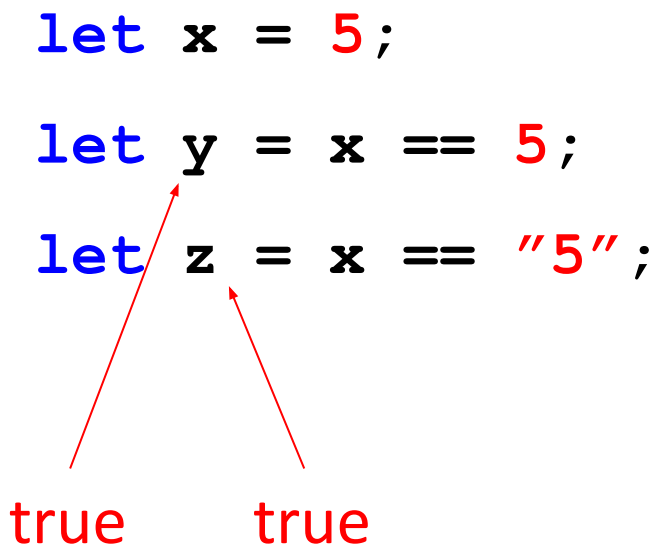
"Hello5"

Toán tử so sánh (comparision)

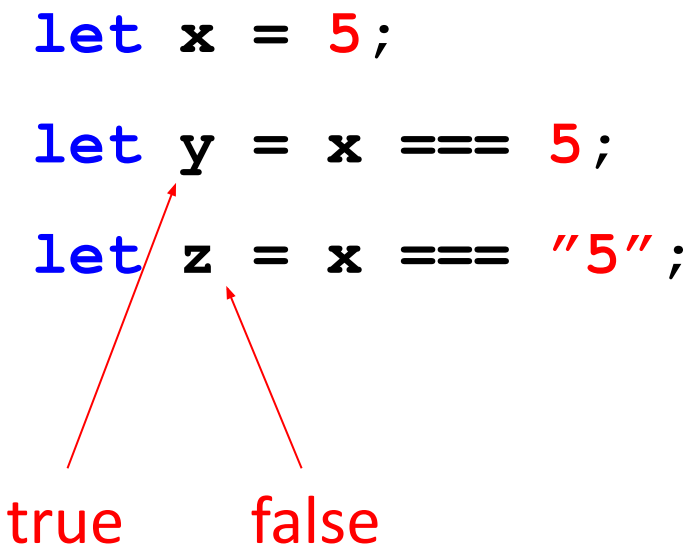
- Toán tử so sánh được dùng để đánh giá mức độ tương quan giữa các giá trị

Operator	Description
==	equal to (bằng)
===	equal value and equal type (bằng giá trị đồng thời cùng kiểu dữ liệu)
!=	not equal (khác)
!==	not equal value or not equal type (không bằng giá trị hoặc không cùng kiểu dữ liệu)
>	greater than (lớn hơn)
<	less than (nhỏ hơn)
>=	greater than or equal to (lớn hơn hoặc bằng)
<=	less than or equal to (nhỏ hơn hoặc bằng)

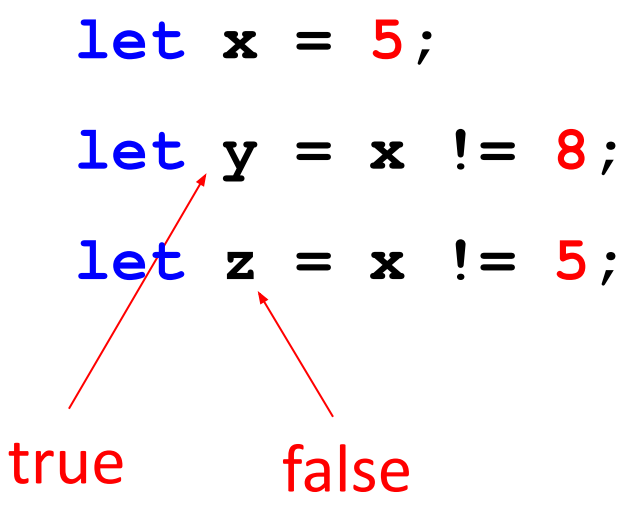
Toán tử so sánh bằng (==): Ví dụ



Toán tử so sánh bằng (===): Ví dụ



Toán tử so sánh khác: Ví dụ



Toán tử logic (logical)

- Toán tử logic được dùng trong các biểu thức logic (true/false)
- && là toán tử ”và”
- || là toán tử “hoặc”
- ! Là toán tử “phủ định”

Operator	Description
&&	logical and
||	logical or
!	logical not

Toán tử &&

Giá trị biến a	Giá trị biến b	Kết quả (a && b)
true	true	true
true	false	false
false	true	false
false	false	false

Toán tử | |

Giá trị biến a	Giá trị biến b	Kết quả (a || b)
true	true	true
true	false	true
false	true	true
false	false	false

Toán tử !

Giá trị biến a	Kết quả !a
true	false
false	true

Toán tử typeof

- Toán tử typeof được dùng để lấy về kiểu dữ liệu của một biến hoặc giá trị

```
typeof "John"                // Returns "string"
typeof 3.14                  // Returns "number"
typeof true                  // Returns "boolean"
typeof false                 // Returns "boolean"
```

Độ ưu tiên của các toán tử

- Trong một biểu thức có nhiều phép toán thì chúng sẽ lần lượt được đánh giá dựa vào độ ưu tiên
- Có thể sử dụng dấu ngoặc "()" để thay đổi độ ưu tiên của các toán tử
- Các toán tử có cùng độ ưu tiên thì sẽ thực hiện từ trái sang phải

Operators	Precedence
postfix	expr++ expr--
unary	++expr --expr +expr -expr ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Độ ưu tiên của các toán tử: Ví dụ

```
let x = 5;
let y = 10;
let z =    (++x  *  y)  <  5   *  10  &&  6  >  3;

           (6    *  y)  <  5   *  10  &&  6  >  3;

           (   60    )  <      50      &&  6  >  3;

                        false           &&  true;

                              false
```

Demo

Toán tử gán
Toán tử số học
Toán tử so sánh
Toán tử logic

Tóm tắt bài học

- Biến được sử dụng để đại diện cho vùng nhớ chứa dữ liệu
- Dữ liệu trong vùng nhớ được gọi là giá trị
- Có thể thay đổi giá trị của biến thông qua phép gán
- Hằng số đại diện cho một giá trị cố định
- Kiểu dữ liệu được sử dụng để phân loại dữ liệu
- Các kiểu dữ liệu thông dụng: chuỗi, số, boolean
- Toán tử được sử dụng để thực hiện các thao tác trong biểu thức
- Các loại toán tử thông dụng: Toán học, gán, cộng chuỗi, so sánh, logic, typeof...
- Các toán tử được thực hiện lần lượt theo độ ưu tiên

Hướng dẫn

Hướng dẫn làm bài thực hành và bài tập

Chuẩn bị bài tiếp theo: *Cấu trúc điều kiện*