1. Giới thiệu

* Hiện nay, Javascript có thể chạy trên cả “Browser” và cả “server”
* Không yêu cầu truy cập mức độ thấp vào bộ nhớ hoặc CPU bởi vì ban đầu ngôn ngữ này được thiết kế cho “Browser”
* Có thể tìm hiểu một số ngôn ngữ liên quan vì đôi khi cần phải sử dụng những tính năng khác nhau
* Các công cụ phát triển cho phép nhìn thấy lỗi, chạy thử, thay đổi các biến,…

1. Kiến thức cơ bản
2. Hello, world
   * Một chương trình JavaScript có thể chèn vào văn bản html bằng thẻ <script> :

Graphical user interface, text, application

Description automatically generated

* + Chúng ta có thể dung thẻ <script> truy xuất một đoạn chương trình ở một file khác. Khi đó, câu lệnh sau đó trong <script> sẽ bị bỏ qua:

Text

Description automatically generated

Lệnh alert(1); sẽ vị bỏ qua.

1. Câu lệnh
   * Lệnh alert() được dùng để in câu lệnh:



Kết quả thu được khi chạy chương trình là :

A picture containing shape

Description automatically generated

* + Dấu ‘;’ được dùng đề kết thúc một câu lệnh, chương trình sẽ coi ‘;’ là dấu hiệu câu lệnh đó đã hết và dịch nó :

Text

Description automatically generated

Sẽ có màn hình thông báo Hello sau đó là 1 và sau đó là 2 nhưng với trường hợp dưới đây chương trình thông báo lỗi :

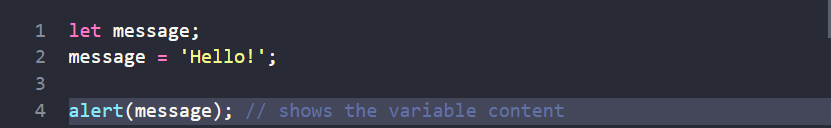
Graphical user interface

Description automatically generated with medium confidence

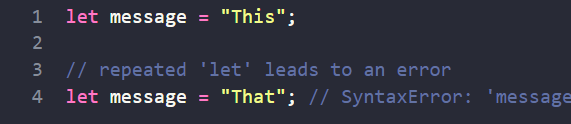
Chương trình trên sẽ được trình dịch hiểu là : alert(“Hello”)[1, 2].forEach(alert);.

* + Chúng ta có thể sử dụng các ghi chú để chú thích công dụng, ý đồ của câu lệnh hoặc một đoạn chương trình
    - ‘//’ chú thích trên một dòng
    - ‘/\*…\*/ chú thích nhiều dòng

1. Chế độ “modern”
   * Chúng ta dùng lệnh ‘use strict’ để kích hoạt chế độ “modern”
   * Câu lệnh này phải được đặt ở đầu chương trình và không có lệnh tương ứng để hủy chế độ “modern”
2. Biến
   * ‘let’ được dùng để khai báo một biến



* + Có thể khai báo nhiều biến một lúc và có thể gán luôn giá trị cho biến.
  + Có thể dùng ‘var’ thay cho ‘let’
  + Giá trị của biến có thể thay đổi nhưng không được khai báo biến quá một lần:

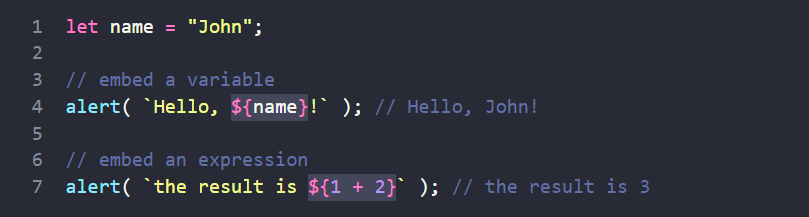


* + Tên của biến chỉ được chứa số, chữ hoặc kí tự ‘$’ hoặc ‘\_’ và không được bắt đầu bằng chữ số
  + JavaScript có phân biệt chữ hoa và chữ thường : apple và APPLE là hai biến khác nhau
  + ‘const’ được dùng để khai báo hằng, giá trị của hằng được giữ nguyên và không thể thay đổi. Để thuận tiện, người ta thường đặt tên hằng bằng chữ cái in hoa:

Text

Description automatically generated

1. Kiểu dữ liệu
   * Có 8 kiểu dữ liệu trong JavaScript
   * Kiểu Number đại diện cho cả số nguyên và số thực có thể được sử dụng trong các toán tử +, - \* /. Ngoài các giá trị số thường gặp còn có giá trị đặc biệt Infinity (1/0), -Infinity và NaN ( lỗi phép tính : “not a nunber” / 2 )
   * BigInt sử dụng cho các số nguyên lớn
   * String là kiểu dữ liệu cho phép lưu trữ một chuỗi các kí tự. Có ba kiểu ngoặc của string “ “ , ‘ ‘, *‘ ‘.* Hai kiểu đầu không khác nhau nhưng kiểu cuối cùng cho phép ta chèn biến hay một biểu thức vào một string sử dùng ${ }.



* + Kiểu Boolean gồm true hoặc false
  + Kiểu “null” thể hiện là giá trị chưa biết, còn trống. Ngoài ra còn có giá trị “undefined"”
  + Object dùng để lưu trữ một kiểu dữ liệu phức tạp ( chứa trong nó nhiều kiểu dữ liệu khác nhau)
  + Symbol dùng cho những định nghĩa mang tính đặc biệt
  + Có thể dùng lệnh ‘typeof x’ để trả về một string là tên kiểu dữ liệu của x :

Text

Description automatically generated

* + Bài tập:

Text

Description automatically generated

3 : hello 1

5: hello name

7: hello Ilya

1. Lệnh tương tác
   * Alert dùng đề hiện thị thông báo và chờ cho tới khi người dùng nhấn “OK” trên modal window
   * Prompt dùng để hiện thị một modal window có thông báo text, một trường để người dùng điền thông tin và nút “OK” hoặc “Cancel”, lệnh này chấp nhận 2 biến:

Text

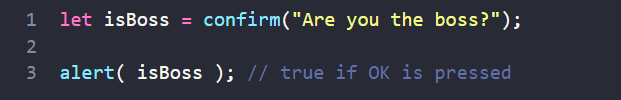
Description automatically generated

Kết quả chạy đoạn chương trình:

Graphical user interface, application

Description automatically generated

* + Confirm dùng để hiện thị modal window với một câu hỏi và nút OK and Cancel. Giá trị trả về sẽ là true nếu chọn OK :



Graphical user interface, application, Teams

Description automatically generated

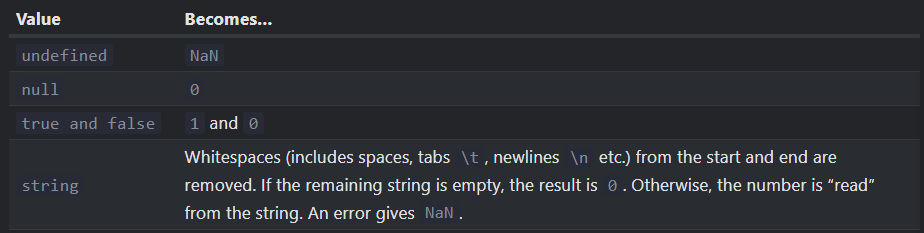
Khi ta chọn OK sẽ có một modal window khác hiện lên với kết quả tương ứng:

Graphical user interface, application, Teams

Description automatically generated

* + Bài tập : let name = prompt(“What is your name?”, ‘ ‘); alert( name);

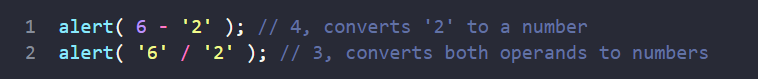
1. Chuyển đổi các kiểu dữ liệu
   * Thông thường lệnh ‘alert’ tự chuyển đổi kiểu dữ liệu cho phù hợp với toán tử
   * Ta có bảng chuyển đổi:

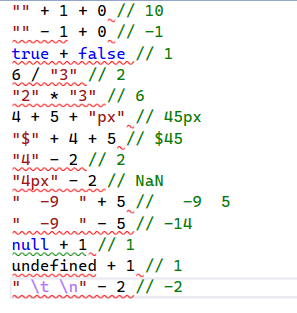


* + Chuyển từ kiểu Boolean càng đơn giản. Các giá trị 0, chuỗi trống, null, undefined và NaN trả về giá trị false còn lại trả về giá trị true

1. Toán tử cơ bản
   * Ngoài toán tử cơ bàn + - \* / còn có % ( chia lấy phần dư) và \*\* ( hàm mũ)
   * Toán tử + có thể cho phép chuyển đổi các toán hạng là số sang kiểu string còn các toán tử khác thì chỉ cho phép chuyển từ string sang số :





* + Ngoài toán tử hai ngôi ở trên ta có toán tử một ngôi ‘+’. Toán tử này hoạt động như Number( ) nhưng ngắn hơn.
  + Thứ tự ưu tiên như trong bảng sau :
  + \
  + Có thể sử dụng toán tử -- hay ++ để giảm đi hay tăng lên 1 giá trị của BIẾN. Vị trí đặt toán tử có liên quan tới trình tự thực hiện phép toán.
  + ‘,’ được dùng để ngăn cách các biểu thức nhưng biểu thức cuối cùng mới được thực hiện
  + Bài tập:
    - 
    - Alert(+a + +b);

1. So sánh
   * Toán tử so sánh < > == != được sử dụng như bình thường với số và giá trị trả về là true hoặc false
   * Khi so sánh string người ta dùng quy luật của bảng chữ cái:

Text

Description automatically generated

* + Khi so sánh các loại dữ liệu khác nhau các kiểu dữ liệu sẽ được chuyển về kiểu number
  + Do == không thể phân biệt 0 với false hay “” nên người ta sử dụng === trong các trường hợp cần thiết
  + Khi thực hiện so sánh với null/undefined cần cẩn thận do đó nên kiểm tra và chuyển về number trước

1. Lệnh điều kiện
   * Text

     Description automatically generated
   * Câu lệnh trong khối sẽ được thực thực hiện khi và chỉ khi điều kiện trong if ( ) đúng
   * Ngoài ra có thể để điều kiện luôn sai (0) hoặc luôn đúng(1)
   * Trong trường hợp có nhiều trường hợp xảy ra thì có thể sử dụng else hoặc if else :

Graphical user interface, text, application, chat or text message

Description automatically generated

* + Toán tử điều kiện ‘?’ có thể dùng để thay thế cho lệnh if :

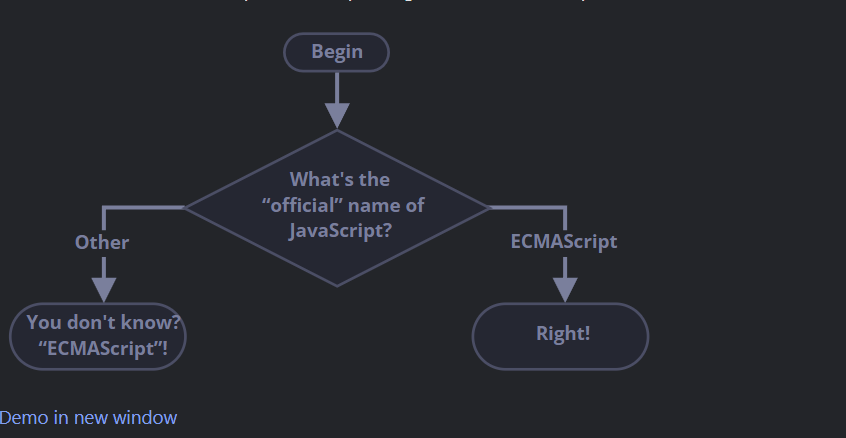


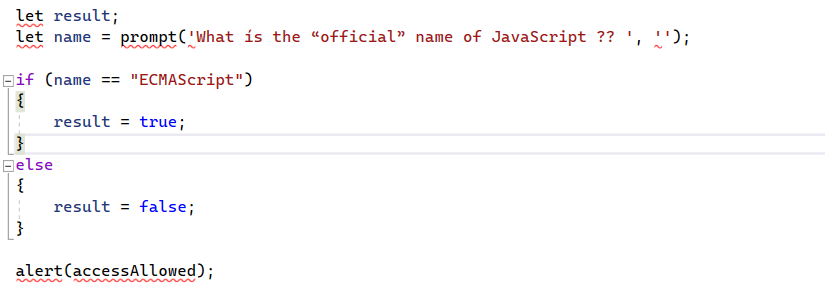
* + Tuy nhiên nó không được khuyến khích vì dễ gây nhầm lẫn
  + Bài tập :

Text

Description automatically generated

* + - Không hiện thị gì



* + - 

1. Toán tử logic
   * || : OR
     + Duyệt từ trái sang phải và dừng lại nếu tìm được giá trị true đầu tiên hoặc trả về giá trị cuối cùng



Nếu tất cả các biến đều bằng không hoặc tương được giá trị false trong Boolean thì Anonymous sẽ được hiện thị

* + && : AND
    - Cũng duyệt từ trái sang phải nhưng sẽ dừng lại nếu gặp giá trị false đầu tiên hoặc trả về giá trị cuối cùng khi tất cả giá trước đó đều true
  + != : NOT
    - Trả về giá trị phủ định của giá trị ban đầu
  + Tất cả những biểu thức chứa toán tử logic thì các toán hạng sẽ được chuyển về dạng boolean trước khi thực hiện phép so sánh
  + Alert trả về giá trị undefined ( =0)
  + Bài tập :



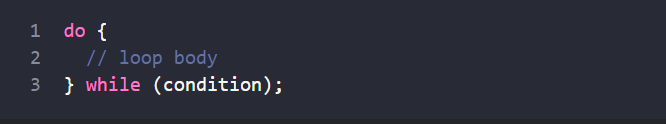
* + - 2

1. Toán tử ‘??’
   * Là một toán tử mới được them vào gần đây, dùng để kiểm tra xem một biến đã được định nghĩa hoặc khác null không ?
   * Kết quả của a??b là a nếu a khác null hoặc đã định nghĩa, ngược lại thì là b.
   * Có thể dùng toán tử này để tìm ra giá trị đầu tiên trong một chuỗi khác null/undefined
   * Khác với toán tử || không thể phân biệt false, 0, “” với null/undefined nhưng ?? có
   * Thứ tự ưu tiên bằng với ||, khi sử dụng cùng các toán tử khác nên sử dụng () để tránh nhầm lẫn
2. Vòng lặp
   * Vòng lặp While, nếu điều kiện còn đúng thì câu lệnh trong vòng lặp sẽ được thực hiện

Text

Description automatically generated with medium confidence

* + Vòng lặp do…while thực hiện câu lệnh trong vòng lặp xong mới kiểm tra điều kiện, có ít nhất một lần thực hiện câu lệnh:



* + Vòng lặp for lặp với số lần biết trước

Text

Description automatically generated

* + Trong một số trường hợp ta cần thoát vòng lặp khi thỏa mãn điều kiện nào đó có thể dùng câu lệnh break:

Graphical user interface, application, website

Description automatically generated

* + Một số trường hợp khác cần thực hiện một vòng lặp mới mà không cần thực hiện tất cả câu lệnh trong vòng lặp thì có dùng lệnh continue:

Text

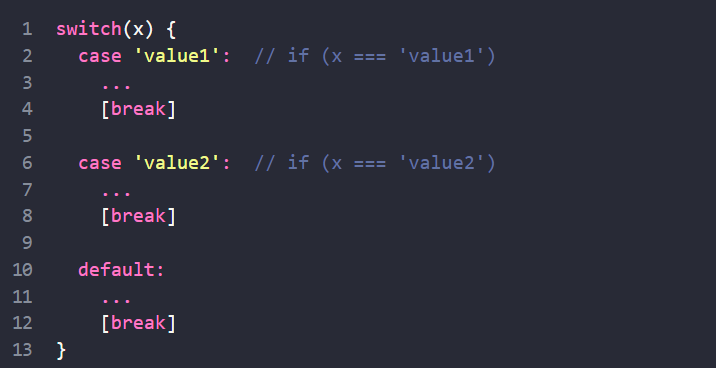
Description automatically generated

* + Để thoát khỏi vòng lặp tùy ý trong một vòng lặp lồng nhau ta có thể sử dụng Label cho vòng lặp và gọi nó cùng lệnh break:

Text

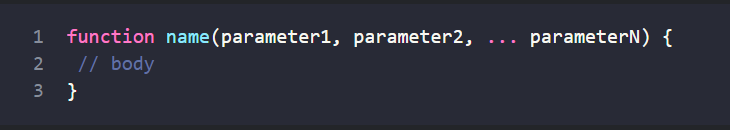
Description automatically generated

1. ‘Switch’
   * Có thể thay thể hàm if phức tạp
   * Có cấu trúc:



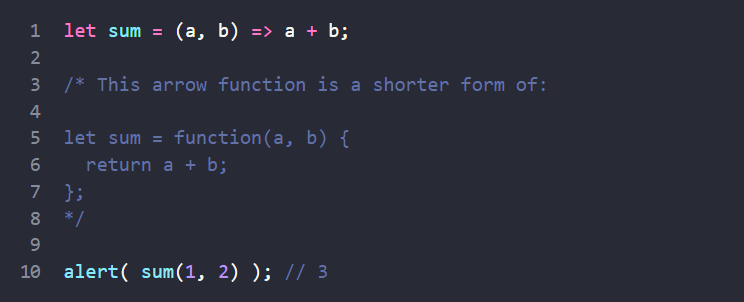
* + Có thể dùng một biểu thức làm ở switch/case
  + Nếu hai hay nhiều case thực hiện cùng một (một số) thao tác thì có thể nhóm các case lại với nhau.

1. Hàm
   * Để khai báo một hàm ta dùng cấu trúc và khi cần gọi hàm ta chỉ cần gọi bằng tên

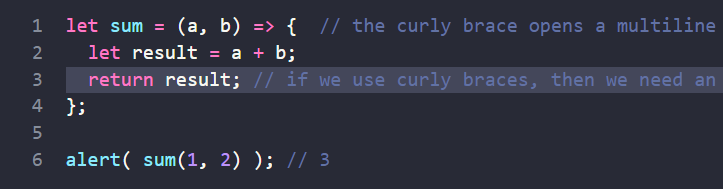


* + Các biến được khai báo trong hàm là các biến cục bộ và chỉ có phạm vi hoạt động trong hàm. Các biến toàn cục được khai báo ngoài hàm cũng có thể được sử dụng trong hàm. Nếu tên biến trong hàm và ngoài hàm trùng nhau thì hàm sẽ sử dụng giá trị của biến khai báo trong hàm.
  + Khi một hàm được gọi nhưng các tham số không được cung cấp đủ thì các tham số thiếu được hiểu là các giá trị undefined
  + Hàm có thể trả về một giá trị, string hoặc không gì cả, khi đó hàm sẽ kết thúc
  + Hàm thường ngắn gọn và chỉ thực hiện một hành động cụ thể, tên của hàm thường được đặt theo chức năng của hàm.

1. Biểu thức hàm
   * Hàm có thể coi là một giá trị và có thể thực hiện các thao tác sao chép, gán giá trị
   * Khai báo biến được thực hiện trước khi chương trình chạy và nên có thể dùng trước khai báo trong cùng một khối
   * Biểu thức hàm chỉ được tạo khi chương trình chạy tới đó nên sử dùng trước khi tạo là sai
2. Hàm Arrow



* + Dòng 1 cho biết sum được gán giá trị của một hàm nhận a,b là tham số và trả về giá trị a+b.
  + Nếu chỉ có một biến thì ta có thể không cần dấu ()
  + Nếu muốn thực hiện một số thao tác ta cần dấu { }. Khi đó, cần trả về một giá trị nhất định đúng với cú pháp:



1. JavaScript specials
2. Code quality
3. Debugging in the browser
   * Debugging là quá trình tìm và sửa lỗi trong script
   * Có thể dùng breakpoint để bắt đầu debugging từ vị trị tùy chọn trong chương trình và lệnh debugger được dùng đề kết thúc quá trình debugging
4. Code Style
   * Syntax:

Graphical user interface

Description automatically generated

Không có quy luật bắt buộc phải theo nhưng được khuyến khích để dễ dàng theo dõi và kiểm tra

1. Comments
   * Được dùng để chú thích: nhiệm vụ chung của chương trình, chức năng của hàm, sự quan trọng của giải thuật
   * Càng ngắn gọn xúc tích càng tốt
2. Ninja code
3. Automated testing with Mocha
4. Polyfills and transpilers