

BÀI 5:

Lập trình kịch bản với JavaScript

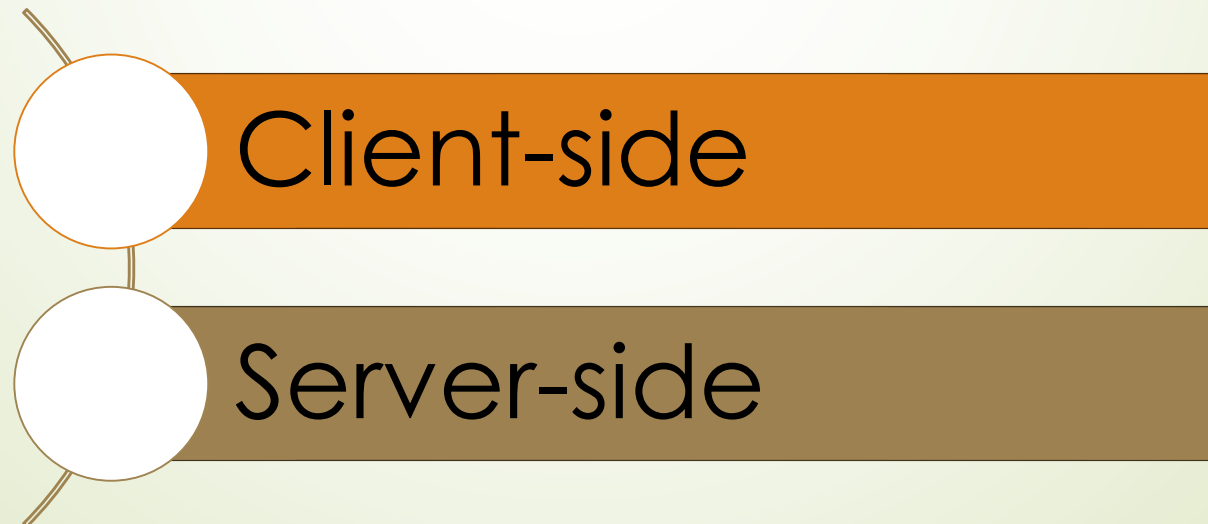


Mục tiêu bài học

- Giới thiệu về JavaScript
- Sử dụng và chèn mã kịch bản JavaScript vào trang web
- Sử dụng các thành phần cú pháp trong JavaScript
 - Biến, kiểu dữ liệu
 - Các toán tử
 - Các cấu trúc điều kiện: If else, switch case
 - Các cấu trúc lặp: for, while, do while
- Định nghĩa và gọi hàm trong JavaScript

◆ Kịch bản (script):

- ◆ Là một khối mã lệnh được tích hợp vào các trang web để giúp cho trang trở nên linh động và tương tác tốt hơn.
- ◆ Có 2 loại kịch bản
 - ◆ Client-side script: Là những mã kịch bản được xử lý và thực thi ở phía máy khách (client). Client-side script được thực thi ngay tại trình duyệt mà không cần phải gửi lên web server.
 - ◆ Server-side script: Là những đoạn mã kịch bản được xử lý và thực thi ở phía máy chủ web. Để thực thi server-side script, ta cần có web server.



- ◆ JavaScript cung cấp những lợi ích sau:
 - ◆ Xử lý các sự kiện cho trang web
 - ◆ JS cho phép người dùng có thể xử lý những tương tác của người dùng trên trang, VD như mouse hover, key v.v... Do đó làm tăng tính tương tác của trang.
 - ◆ Thu thập thông tin về trình duyệt
 - ◆ JS cho phép lấy ra thông tin về trình duyệt, cũng như các đối tượng của trình duyệt, để thao tác khi cần thiết
 - ◆ Xử lý, tính toán cho trang
 - ◆ JS cho phép thực hiện những tính toán đơn giản ngay tại trình duyệt mà không cần phải gửi dữ liệu lên server
 - ◆ Kiểm tra dữ liệu cho form
 - ◆ JS cho phép thực hiện kiểm tra dữ liệu nhập của người dùng trước khi gửi dữ liệu lên server, do đó làm giảm bớt lượng dữ liệu gửi lên server để kiểm tra.

◆ JavaScript:

- ◆ Là một ngôn ngữ kịch bản, cho phép tạo ra những đoạn mã kịch bản được nhúng vào trang web.
- ◆ Các đoạn mã kịch bản JS có thể được nhúng trực tiếp vào trang web bằng cách viết mã nằm bên trong cặp thẻ `<SCRIPT></SCRIPT>`.
- ◆ Có 2 cách sử dụng mã JS
 - ◆ Nhúng trực tiếp vào trang web bên trong thẻ `<SCRIPT>`, trong phần `<HEAD>` hoặc `<BODY>`
 - ◆ Viết mã ở một file JS bên ngoài rồi nhúng file JS vào trang web. File JS có phần mở rộng là `.js`.
 - ◆ Sử dụng thuộc tính `src` để chỉ định đường dẫn của file external JS bên ngoài.

◆ Chèn mã JavaScript:

- ◆ Có thể chèn vào những vùng dưới đây của trang web bằng cách sử dụng thẻ `<SCRIPT>`:

Head

- Đoạn mã kịch bản sẽ được thực thi để đáp ứng một hành động được thực hiện bởi người dùng, hoặc được tự động thực thi.

Body

- Đoạn mã kịch bản sẽ được thực thi ngay khi trang được load

- ◆ Mã kịch bản JS có thể được nhúng vào trang web thông qua cú pháp sau:
`<SCRIPT type="text/javascript"> JavaScript statements
</SCRIPT>`

- ◆ Một số quy tắc viết mã của JavaScript:
 - ◆ JavaScript phân biệt chữ hoa-thường
 - ◆ Mỗi câu lệnh phải được kết thúc bởi dấu (;)
 - ◆ Các chuỗi trong JS có thể được đặt trong dấu “ hoặc ”
 - ◆ Để khai báo biến, ta sử dụng từ khóa var. Không cần phải khai báo kiểu dữ liệu cho biến.
 - ◆ Để nối chuỗi, ta sử dụng dấu +

◆ Biến:

- ◆ Là một vị trí trong bộ nhớ máy tính được đặt tên, dùng để lưu trữ giá trị trong các chương trình.
- ◆ Trong JS, ta khai báo biến bằng cách sử dụng từ khóa var.
- ◆ Để khai báo biến, ta sử dụng cú pháp sau:
`var var_name;`
- ◆ VD: `var tuoi = 20;`

◆ Toán tử:

- ◆ Là các phép tính được dùng cho việc tính toán hoặc so sánh giá trị trên các toán hạng.
- ◆ Có thể được dùng để thay đổi giá trị của các biến.
- ◆ JS hỗ trợ đầy đủ các toán tử, được phân chia thành các loại sau:

Arithmetic

Assignment

Arithmetic
Assignment

Comparison

Logical

◆ Toán tử số học (arithmetic operator):

- ◆ Được dùng để thực hiện các phép tính số học trên các biến và literals.
- ◆ Bao gồm các phép tính sau:
 - ◆ +
 - ◆ -
 - ◆ *
 - ◆ /
 - ◆ %

◆ Toán tử gán (assignment operator):

- ◆ Được dùng để gán giá trị hoặc một kết quả của một biểu thức ở phía bên phải cho một biến hay biểu thức ở phía bên trái.

◆ Toán tử gán đại số (Arithmetic assignment operators):

- ◆ Dùng để thực hiện các phép tính đại số và gán giá trị cho biến ở phía bên trái của toán tử.
- ◆ Bao gồm các phép tính sau:
 - ◆ $+=$
 - ◆ $-=$
 - ◆ $*=$
 - ◆ $/=$
 - ◆ $\%=$

◆ Toán tử quan hệ (so sánh)

- ◆ Được dùng để so sánh 2 giá trị và thực hiện một hành động dựa trên kết quả của phép so sánh.
- ◆ Gồm có những toán tử sau:

- ◆ <
- ◆ >
- ◆ <=
- ◆ >=
- ◆ ==
- ◆ !=
- ◆ ===

◆ Toán tử luận lý (Logical operators):

- ◆ Được dùng để đánh giá những biểu thức luận lý (logic).
- ◆ Kết quả trả về là một giá trị Boolean: true hoặc false.
- ◆ Gồm những toán tử sau:
 - ◆ `&&`: Chỉ trả về true nếu cả hai toán hạng đều là true. Sẽ trả về false nếu một trong hai toán hạng là false.
 - ◆ `!`: Trả về giá trị phủ định của toán hạng.
 - ◆ `||`: Chỉ trả về false nếu cả hai toán hạng đều là false. Sẽ trả về true khi chỉ cần một trong hai toán hạng là true.

◆ Cấu trúc điều kiện

- ◆ Là những cấu trúc lập trình cho phép ta thực thi một khối lệnh dựa trên kết quả của biểu thức được đánh giá.
- ◆ Có 2 cấu trúc điều kiện như sau:

`if...else`

`switch...case`

◆ Cấu trúc if

- ◆ Dùng để kiểm tra một điều kiện. Nếu điều kiện thỏa mãn thì sẽ thực hiện thân của khối if. Nếu điều kiện không thỏa mãn thì bỏ qua khối if, chương trình thực hiện câu lệnh sau if.
- ◆ Biểu thức điều kiện của if là một biểu thức quan hệ hoặc logic, và kết quả trả về là một giá trị logic true hoặc false.
- ◆ Cú pháp:

```
if (condition) {  
    // block of code to be executed if the condition is true  
}
```

◆ Ví dụ:

```
var age = 20;  
  
if( age > 18 ) {  
    document.write("<b>Qualifies for driving</b>");  
}
```

Cấu trúc điều kiện

◆ Cấu trúc `if...else`:

- ◆ Là cú pháp đầy đủ hơn của cấu trúc `if`.
- ◆ Nếu điều kiện thỏa mãn thì chương trình thực hiện khối `if`. Nếu điều kiện không thỏa mãn thì thực hiện khối `else`.
- ◆ Cú pháp:

```
if (expression) {  
    Statement(s) to be executed if expression is true  
} else {  
    Statement(s) to be executed if expression is false  
}
```

◆ Ví dụ:

```
var age = 15;  
  
if( age > 18 ) {  
    document.write("<b>Qualifies for driving</b>");  
} else {  
    document.write("<b>Does not qualify for driving</b>");  
}
```

◆ Cấu trúc if...elseif:

- ◆ Là cú pháp mở rộng của cấu trúc if else.
- ◆ Được sử dụng trong trường hợp chương trình có nhiều điều kiện cần đánh giá.
- ◆ Cú pháp:

```
if (expression 1) {  
    Statement(s) to be executed if expression 1 is true  
} else if (expression 2) {  
    Statement(s) to be executed if expression 2 is true  
} else if (expression 3) {  
    Statement(s) to be executed if expression 3 is true  
} else {  
    Statement(s) to be executed if no expression is true  
}
```

◆ Ví dụ:

```
<html>  
  <body>  
    <script type = "text/javascript">  
      <!--  
        var book = "maths";  
        if( book == "history" ) {  
          document.write("<b>History Book</b>");  
        } else if( book == "maths" ) {  
          document.write("<b>Maths Book</b>");  
        } else if( book == "economics" ) {  
          document.write("<b>Economics Book</b>");  
        } else {  
          document.write("<b>Unknown Book</b>");  
        }  
      //-->  
    </script>  
    <p>Set the variable to different value and then try...</p>  
  </body>  
</html>
```

Cấu trúc switch case

◆ Cấu trúc switch case:

- ◆ Cho phép đánh giá giá trị của một biểu thức, và nhiều khối lệnh được thực hiện dựa vào giá trị của biểu thức. Nếu không có khối lệnh nào được thực hiện, khối default sẽ được thực thi.
- ◆ Cơ chế thực hiện tương tự như cấu trúc if ... else if
- ◆ Chú ý: Kết thúc mỗi nhánh case cần có lệnh break, để kết thúc case đó.
- ◆ Cú pháp:

```
switch (expression) {  
    case condition 1: statement(s)  
        break;  
  
    case condition 2: statement(s)  
        break;  
    ...  
  
    case condition n: statement(s)  
        break;  
  
    default: statement(s)  
}
```


Cấu trúc switch case

◆ Ví dụ:

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var grade = 'A';
        document.write("Entering switch block<br />");
        switch (grade) {
          case 'A': document.write("Good job<br />");
                     break;

          case 'B': document.write("Pretty good<br />");
                     break;

          case 'C': document.write("Passed<br />");
                     break;

          case 'D': document.write("Not so good<br />");
                     break;

          case 'F': document.write("Failed<br />");
                     break;

          default:  document.write("Unknown grade<br />")
        }
        document.write("Exiting switch block");
      <!-->
    </script>
    <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

◆ Kết quả:


```
Entering switch block
Good job
Exiting switch block
Set the variable to different value and then try...
```

Cấu trúc điều kiện

- GV minh họa code và sinh viên thực hành cấu trúc điều kiện

◆ Cấu trúc lặp

- ◆ Là những cấu trúc cho phép thực hiện lặp đi lặp lại một khối lệnh nhiều lần.
- ◆ Các vòng lặp sẽ có biểu thức điều kiện. Điều kiện của vòng lặp là một biểu thức quan hệ hoặc logic.
- ◆ Các vòng lặp sẽ được thực hiện chừng nào điều kiện còn thỏa mãn.
- ◆ Có 3 loại vòng lặp như sau:



while

do...while

for

◆ Vòng lặp `while`:

- ◆ Là vòng lặp sẽ được thực hiện chừng nào điều kiện còn thỏa mãn (trả về true).
- ◆ Vòng lặp `while` sẽ kiểm tra điều kiện trước, sau đó mới thực thi thân của vòng lặp.
- ◆ Cú pháp:

```
while (expression) {  
    Statement(s) to be executed if expression is true  
}
```

Giới thiệu về cấu trúc lặp

◆ Ví dụ:

```
<html>
  <body>

    <script type = "text/javascript">
      <!--
        var count = 0;
        document.write("Starting Loop ");

        while (count < 10) {
          document.write("Current Count : " + count + "<br />");
          count++;
        }

        document.write("Loop stopped!");
      <!-->
    </script>

    <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

◆ Kết quả:

```
Starting Loop
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Current Count : 5
Current Count : 6
Current Count : 7
Current Count : 8
Current Count : 9
Loop stopped!
Set the variable to different value and then try...
```

◆ Vòng lặp do while:

- ◆ Là vòng lặp sẽ được thực hiện chừng nào điều kiện còn thỏa mãn (trả về true).
- ◆ Vòng lặp while sẽ thực thi thân của vòng lặp trước, sau đó mới lặp kiểm tra điều kiện. Do đó nếu điều kiện sai, vòng lặp do while sẽ được thực hiện ít nhất một lần.
- ◆ Cú pháp:

```
do {  
    Statement(s) to be executed;  
} while (expression);
```

Giới thiệu về cấu trúc lặp

♦ Ví dụ:

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var count = 0;

        document.write("Starting Loop" + "<br />");
        do {
          document.write("Current Count : " + count + "<br />");
          count++;
        }

        while (count < 5);
        document.write ("Loop stopped!");
      //-->
    </script>
    <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

♦ Kết quả:

```
Starting Loop
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Loop Stopped!
Set the variable to different value and then try...
```


◆ Vòng lặp for:

- ◆ Là loại vòng lặp được sử dụng khi ta đã biết trước số lần lặp.
- ◆ Vòng lặp for bao gồm 3 phần như sau:
 - ◆ Lệnh khởi tạo: Dùng để khởi tạo biến đếm được sử dụng trong vòng lặp. Được thực hiện một lần trước khi bắt đầu vòng lặp.
 - ◆ Điều kiện: Được kiểm tra trước mỗi lần lặp. Nếu điều kiện thỏa mãn thì sẽ tiếp tục thực hiện vòng lặp. Nếu điều kiện không thỏa mãn sẽ kết thúc vòng lặp.
 - ◆ Lệnh tăng/giảm: Dùng để tăng/giảm giá trị của biến đếm sau mỗi lần lặp.
- ◆ Mỗi phần được phân tách bởi dấu (;). Cả 3 phần đều là tùy chọn.
- ◆ Cú pháp:

```
for (initialization; test condition; iteration statement) {  
    Statement(s) to be executed if test condition is true  
}
```


◇ Ví dụ:

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var count;
        document.write("Starting Loop" + "<br />");

        for(count = 0; count < 10; count++) {
          document.write("Current Count : " + count );
          document.write("<br />");
        }
        document.write("Loop stopped!");
      //-->
    </script>
    <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

◇ Kết quả:

```
Starting Loop
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Current Count : 5
Current Count : 6
Current Count : 7
Current Count : 8
Current Count : 9
Loop stopped!
Set the variable to different value and then try...
```

Lệnh Break & Continue

◆ Lệnh break:

- ◆ Được dùng để kết thúc vòng lặp.
- ◆ Thường được sử dụng bên trong một cấu trúc if để xử lý một điều kiện trong vòng lặp.
- ◆ Ví dụ:

```
<html>
<body>
  <script type = "text/javascript">
    <!--
    var x = 1;
    document.write("Entering the loop<br /> ");

    while (x < 20) {
      if (x == 5) {
        break;    // breaks out of loop completely
      }
      x = x + 1;
      document.write( x + "<br />");
    }
    document.write("Exiting the loop!<br /> ");
    <!-->
  </script>

  <p>Set the variable to different value and then try...</p>
</body>
</html>
```

◆ Kết quả:

```
Entering the loop
2
3
4
5
Exiting the loop!
Set the variable to different value and then try...
```

Lệnh Break & Continue

◆ Lệnh continue:

- ◆ Bỏ qua những câu lệnh còn lại của lần lặp hiện tại, rồi thực hiện lần lặp tiếp theo trong vòng lặp.

◆ Ví dụ:

```
<html>
<body>
  <script type = "text/javascript">
    <!--
      var x = 1;
      document.write("Entering the loop<br /> ");

      while (x < 10) {
        x = x + 1;

        if (x == 5) {
          continue; // skip rest of the loop body
        }
        document.write( x + "<br />");
      }
      document.write("Exiting the loop!<br /> ");
    <!-->
  </script>
  <p>Set the variable to different value and then try...</p>
</body>
</html>
```

◆ Kết quả:

```
Entering the loop
2
3
4
6
7
8
9
10
Exiting the loop!
Set the variable to different value and then try...
```

◆ Hàm (function)

- ◆ Là một nhóm các câu lệnh được đóng gói thành một đơn vị thực thi, dùng để thực hiện một công việc cụ thể.
- ◆ Hàm cho phép tái sử dụng mã nguồn. Mỗi hàm được định nghĩa một lần, sau đó có thể được gọi nhiều lần.
- ◆ Hàm giúp module hóa chương trình. Một chương trình lớn có thể được phân chia thành các hàm, mỗi hàm thực hiện một tác vụ độc lập.
- ◆ JavaScript cung cấp 2 loại hàm:
 - ◆ Built-in functions: Là những hàm được định nghĩa sẵn bởi JS
 - ◆ User-defined functions: Là những hàm do người dùng tự định nghĩa.

Built-in

User-defined

Định nghĩa hàm trong JS

◆ Định nghĩa hàm:

- ◆ Trước khi có thể sử dụng hàm, ta cần định nghĩa hàm.
- ◆ Để định nghĩa hàm, ta sử dụng từ khóa function, theo sau là tên hàm, và một danh sách các tham số trong cặp dấu ()
- ◆ Tiếp theo là thân của hàm, được chứa trong cặp dấu {}
- ◆ Chú ý: Tên hàm phải là duy nhất.

◆ Cú pháp:

```
<script type = "text/javascript">
  <!--
    function functionname(parameter-list) {
      statements
    }
  //-->
</script>
```

Định nghĩa hàm trong JS

◇ Ví dụ:

```
<html>
  <head>
    <script type = "text/javascript">
      function sayHello() {
        document.write ("Hello there!");
      }
    </script>
  </head>

  <body>
    <p>Click the following button to call the function</p>
    <form>
      <input type = "button" onclick = "sayHello()" value = "Say Hello">
    </form>
    <p>Use different text in write method and then try...</p>
  </body>
</html>
```


◆ Đặc điểm của hàm:

- ◆ Để trả về giá trị từ hàm, ta sử dụng từ khóa return.
- ◆ Hàm không tự thực thi mà phải được gọi.
- ◆ Hàm có thể có những tham số. Tham số cho phép thay đổi nội dung thực thi bên trong hàm.

Ví dụ

Ví dụ:

```
<script type="text/javascript">
function add()
{
    var num1=parseInt(prompt("Nhap so thu nhat:"));
    var num2=parseInt(prompt("Nhap so thu hai:"));

    var result=num1+num2;
    alert("Tong hai so: "+result);
}

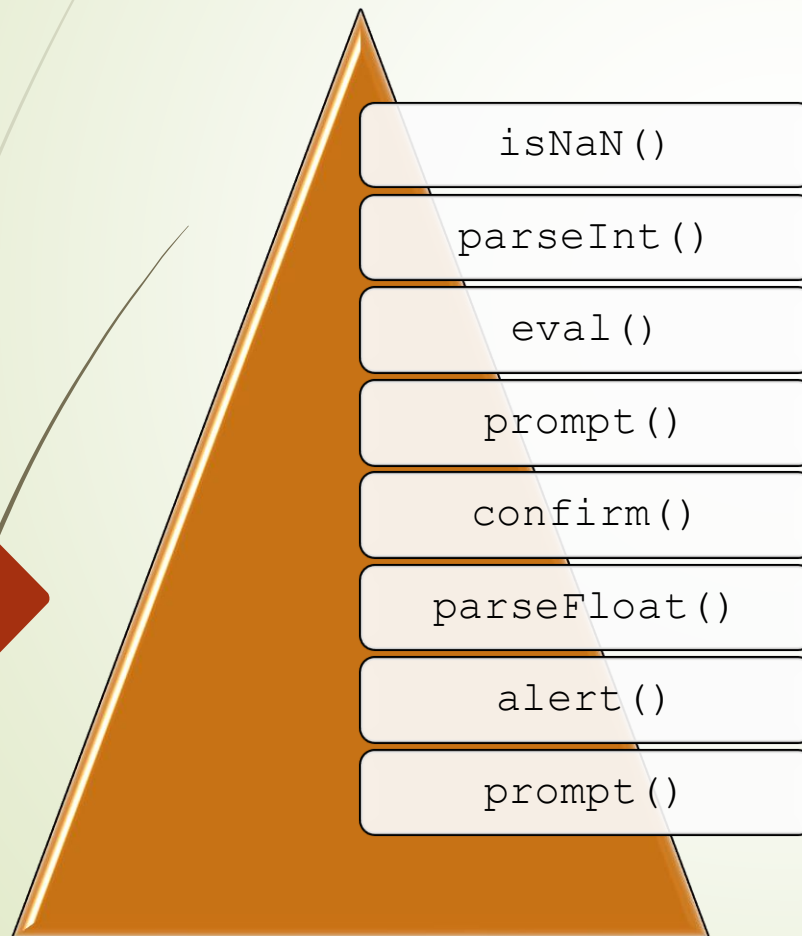
function calling_add()
{
    add();
}

calling_add();
</script>
```

Định nghĩa hàm trong JS

◆ Built-in functions:

- ◆ Là những hàm được cung cấp sẵn bởi JS
- ◆ Danh sách một số hàm thông dụng:



Demo

36

- Giảng viên demo định nghĩa và gọi hàm trong JS.
- Học viên thực hành về hàm trong JS

TÓM TẮT BÀI HỌC

- Giới thiệu về JavaScript
- Sử dụng và chèn mã kịch bản JavaScript vào trang web
- Sử dụng các thành phần cú pháp trong JavaScript
 - Biến, kiểu dữ liệu
 - Các toán tử
 - Các cấu trúc điều kiện: If else, switch case
 - Các cấu trúc lặp: for, while, do while
- Định nghĩa và gọi hàm trong JavaScript