Web Application Development

Javascript

Contents

- ❏ Introduction to JavaScript
- ❏ How to add JavaScript to HTML
- ❏ JavaScript – basic
- ❏ JavaScript – events
- ❏ JavaScript – HTML DOM

Contents

Concept of Script

- **Client-Side Script**
  - Client-side scripting involves executing scripts, such as JavaScript, directly within a web browser (*Client-Side*).
  - Allows web pages to change content dynamically in response to user input or other events.
  - Enables real-time feedback and interactivity without needing to reload the entire webpage
- **Server-Side Script**
  - Server-side scripting involves executing scripts on a web server (*Server-Side*) in response to client requests. These scripts generate dynamic content, often tailored to individual users, which is then sent to the client's web browser to be displayed.

Introduction to JavaScript

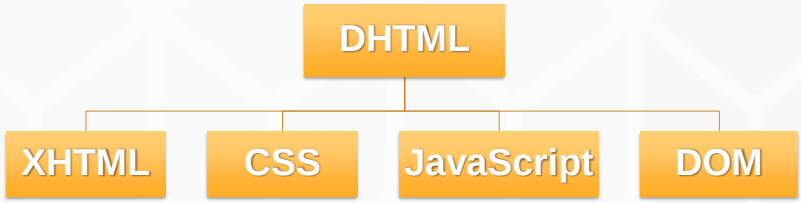
- **JavaScript** is a programming language that allows you to implement complex functionalities on web pages. Every time a web page does more than just sit there and display static information for you to look at—displaying timely content updates, interactive maps, animated 2D/3D graphics, scrolling video jukeboxes, or more—you can bet that JavaScript is probably involved..
- **JavaScript** is a scripting language that enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else.

What can JavaScript really do?

- The core client-side JavaScript language consists of some common programming features that allow you to do things like:
  - Store useful values inside variables
  - Operations on pieces of text (known as "strings" in programming)
  - Running code in response to certain events occurring on a web page
  - And much more!

Dynamic HTML (DHTML)

- **DHTML** refers to a combination of technologies used together to create interactive and dynamic web page content. The main technologies are HTML, CSS, and JavaScript.
- **DHTML** = HTML + CSS + JavaScript



Interpreted versus compiled code

- JavaScript is a lightweight interpreted programming language.
- In interpreted languages, the code is run from top to bottom and the result of running the code is immediately returned. You don't have to transform the code into a different form before the browser runs it. The code is received in its programmer-friendly text form and processed directly from that.
- Compiled languages on the other hand are transformed (compiled) into another form before they are run by the computer

Contents

How to add JavaScript to HTML?

- JavaScript is applied to your HTML page in a similar manner to CSS.
- Internal JavaScript

```
<script type="text/javascript">
  <!--
  // Javascript code
  -->
</script>
```

- External JavaScript

```
<script src="scripts.js"></script>
```

Example

```
<html>
    <head>
        <script type="text/javascript">
                some statements
        </script>
    </head>
    <body>
        <script type="text/javascript">
                some statements
        </script>

        <script src="Tên_file_script.js">method()</script>

        <script type="text/javascript">
                // gọi thực hiện các phương thức được định nghĩa
                // trong "Tên_file_script.js"
        </script>
    </body>
</html>
```

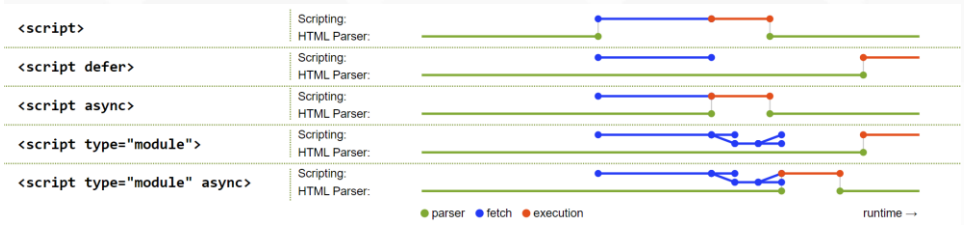
Script loading strategies

- There are a number of issues involved with getting scripts to load at the right time. A common problem is that all the HTML on a page is loaded in the order in which it appears. If you are using JavaScript to manipulate elements on the page, your code won't work if the JavaScript is loaded and parsed before the HTML you are trying to do something to.
- An old-fashioned solution to this problem used to be to put your script element right at the bottom of the body (e.g. just before the </body> tag), so that it would load after all the HTML has been parsed.

Script loading strategies (cont.)

- Scripts loaded using the async attribute will download the script without blocking the page while the script is being fetched.
- Scripts loaded with the defer attribute will load in the order they appear on the page. They won't run until the page content has all loaded, which is useful if your scripts depend on the DOM being in place (e.g. they modify one or more elements on the page).

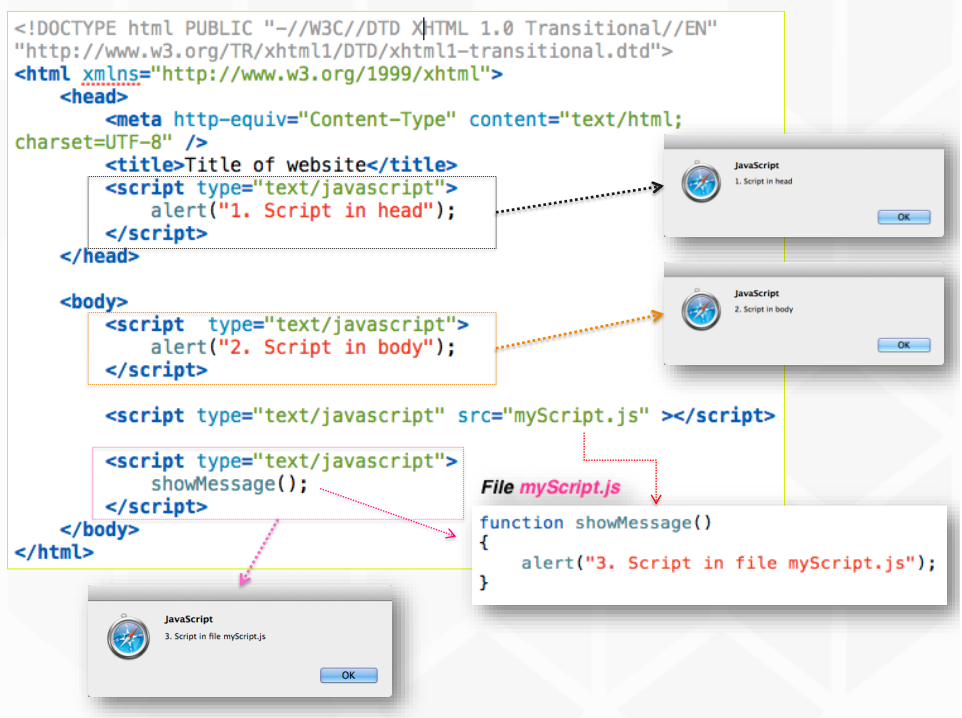
Script loading strategies (cont.)



- Ex.

```
<script async
src="js/vendor/jquery.js"></script>

<script async
src="js/script2.js"></script>

<script async
src="js/script3.js"></script>
```

```
<script defer
src="js/vendor/jquery.js"></script>

<script defer
src="js/script2.js"></script>

<script defer
src="js/script3.js"></script>
```



Contents

- ❏ *Introduction to JavaScript*
- ❏ *How to add JavaScript to HTML*
- ❏ **JavaScript – basic**
- ❏ JavaScript – events
- ❏ JavaScript – HTML DOM

JavaScript Variables

- All JavaScript variables must be identified with unique names. The general rules for constructing names for variables (unique identifiers) are:
  - Names can contain letters, digits, underscores, and dollar signs.
  - Names must begin with a letter.
  - Names can also begin with $ and _ (but we will not use it in this tutorial).
  - Names are case sensitive (y and Y are different variables).
  - Reserved words (like JavaScript keywords) cannot be used as names.

Declaring a JavaScript Variable

```
var v1;          // v1 = undefined
let v2, v3 = 10;// v2 = undefined
const v4 = 'v';
```

- A variable declared without a value will have the value undefined
- If you re-declare a JavaScript variable declared with var, it will not lose its value. You cannot re-declare a variable declared with let or const.
- Variables declared with var are Function-scoped. But if declared outside any function, they have Global Scope.
- Variables defined with let have Block Scope.

JavaScript Data Types

Datatype	Example	Description
Object	const obj = {fName: 'John', lName: 'Doe'};	Objects are used to store key/value (name/value) collections
String	"The cow jumped over the moon." '40'	A string (or a text string) is a series of characters. Strings are written with quotes.
Number	0.066218 12	According to the IEEE 754 standard
Boolean	true / false	
Undefined	let myVariable ;	Any variable can be emptied, by setting the value to undefined. The type will also be undefined.
Null	connection.Close();	connection = null

JavaScript has dynamic types. This means that the same variable can be used to hold different data types

Data type conversion

- In JavaScript, variables can automatically convert between data types depending on the value assigned to them.

  Ex. :

  ```
  let x = 10;              // Number
  x = "hello world !";     // String
  ```

- When you add a number and a string, the number is coerced to a string.

  Ex. :

  ```
  let x;
  x = "12" + 34.5;         // result: x = "1234.5"
  ```

- **parseInt(…)**, **parseFloat(…)** : Parses a string an returns a number.

JavaScript Functions

- Syntax:

```
function function_name(argName1, argName2, ...){
    ...
}
```

- Function with return value :

```
function function_name(argName1, argName2, ...){
    ...
    return <value>;
}
```

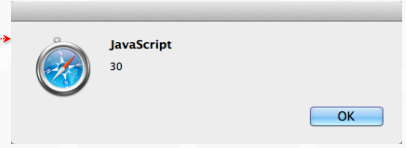
Example

- Declare Function:

```
function Sum(x, y)
{
    tong = x + y;
    return tong;
}
```

- Invoking Function:

```
var x = Sum(10, 20);
alert(x);
```



Function scope and conflicts

- The top level outside all your functions is called the global scope. Values defined in the global scope are accessible from everywhere in the code.
- External scripts that you call in from elsewhere could start to mess with your code and cause problems because they happen to be using the same variable names as other parts of the code, causing conflicts
- Keeping parts of your code locked away in functions avoids such problems, and is considered the best practice.

if, else

```
if (condition)
{
    statement[s] if true
}
else
{
    statement[s] if false
}
```

Ex.:

```
let x = 5, y = 6, z;
if (x === 5)
{
        if (y === 6)
                z = 17;
}
else
        z = 20;
```

Switch Statement

```
switch (expression)
{
    case label1 :
       statementlist
    case label2 :
        statementlist
       ...
   default :
     statement list
}
```

Ex. :

```
let diem = "G";
switch (diem) {
        case "Y":
                document.write("Yếu");
                break;
        case "TB":
                document.write("Trung bình");
                break;
        case "K":
                document.write("Khá");
                // break;
        case "G":
                document.write("Giỏi");
                break;
        default:
                document.write("Xuất sắc");
}
```

For Loop

```
for ([initial expression]; [condition]; [update expression])
{
    statement[s] inside loop
}
```

Ex. :

```
var myarray = new Array();
for (i = 0; i < 10; i++)
{
        myarray[i] = i;
}
```

for...in

```
for (key in object)
{
    statement[s] inside loop
}
```

Ex. :

```
const person = { fname: "A", lname: "Tran", age: 20 };
for (let e in person)
{
        //…
}
```

for...of

```
for (variable of iterable)
{
    statement[s] inside loop
}
```

Ex. :

```
const arr = [ "A", "B", "C" ];
for (let s in arr)
{
        //…
}
```

while & do … while

while *(expression)*
{
    statements
}

Ex. :

```
var i = 9, total = 0;
while (i < 10)
{
        total += i * 3 + 5;
        i = i +5;
}
```

do
{
    statements
} ***while*** *(expression);*

Ex. :

```
var i = 9, total = 0;
do
{
        total += i * 3 + 5;
        i = i +5;
} while (i < 10);
```

Contents

JavaScript Events

- HTML events are "things" that happen to HTML elements. An HTML event can be something the browser does, or something a user does.
- Here are some examples of HTML events:
  - An HTML web page has finished loading
  - An HTML input field was changed
  - An HTML button was clicked

Common HTML Events

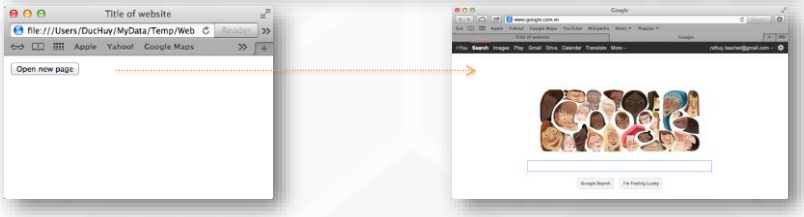
Event	Occurs When	Belongs To
onchange	The content of a form element has changed	Event
onclick	An element is clicked on	MouseEvent
ondblclick	An element is double-clicked	MouseEvent
onfocus	An element gets focus	FocusEvent
onblur	An element loses focus	FocusEvent
onmousedown	The mouse button is pressed over an element	MouseEvent
onmousemove	The pointer is moved over an element	MouseEvent
onmouseup	A user releases a mouse button over an element	MouseEvent
onkeydown	A key is down	KeyboardEvent
onload	An object has loaded	UiEvent, Event
onsubmit	A form is submitted	Event

JavaScript Event Handlers

- You can assign your own event handler functions to HTML elements :

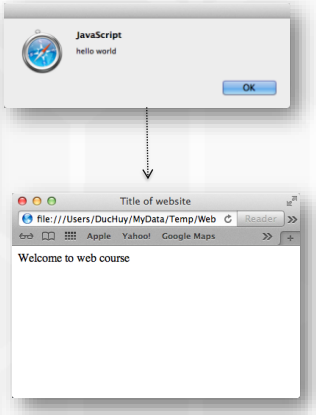
<TAG eventHandler = "JavaScript Code">

```
<body>
    <input  type="button"
            name="btnClickMe"
            value="Open new page"
            onclick="window.open('http://www.gooogle.com');" />
</body>
```

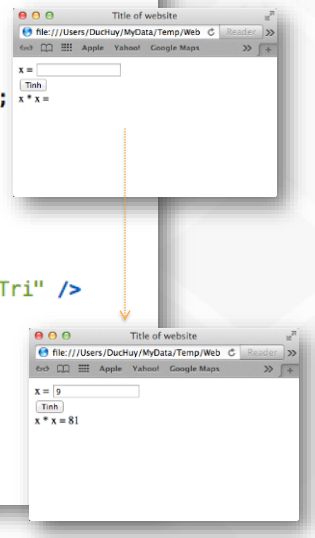


JavaScript Event Handlers (cont.)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type"
content="text/html; charset=UTF-8" />
        <title>Title of website</title>
        <script type="text/javascript">
            function ShowMessage() {
                alert("hello world");
            }
        </script>
    </head>

    <body onload="ShowMessage()">
        Welcome to web course
    </body>
</html>
```



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//
EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />
        <title>Title of website</title>
        <script type="text/javascript">
            function TinhToan() {
                var x = frmMain.txtGiaTri.value;
                ketQua.innerHTML = x * x;
            }
        </script>
    </head>

    <body>
        <form name="frmMain">
            x = <input type="text" name="txtGiaTri" />
            <br />
            <input type="button" value="Tinh"
                   onclick="TinhToan()" />
            <br />
            x * x = <span id="ketQua"></span>
        </form>
    </body>
</html>
```



JavaScript Event Handler Properties

- Objects (such as buttons) that can fire events also usually have properties whose name is on followed by the name of the event.

  object.onevent = function_name;

```
<html>
    <head>
        <script language="Javascript">
            function GreetingMessage()
            {
                window.alert("Welcome to my world");
            }

            window.onload = GreetingMessage ();
        </script>
    </head>

    <body>
    </body>
<html>
```

Using addEventListener()

- Objects that can fire events have an addEventListener() method, and this is the recommended mechanism for adding event handlers.

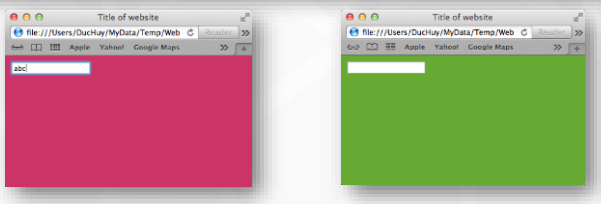
```
window.addEventListener('load', GreetingMessage);
```

- If you've added an event handler using addEventListener(), you can remove it again using the removeEventListener() method.

```
window.removeEventListener('load', GreetingMessage);
```

Example onFocus - onBlur

```
<body>
    <form name="frmMain">
        <input type="text"
               onfocus="document.bgColor='#cc3366';"
               onblur="document.bgColor='#66aa33';"
        />
    </form>
</body>
```



Contents

JavaScript HTML DOM

- **DOM** = **D**ocument **O**bject **M**odel
- When a web page is loaded, the browser creates a Document Object Model of the page. The HTML DOM model is constructed as a tree of Objects
- Some DOM objects: *window, document, element, attribute, text node, history, link, form, frame, location, event, …*

DOM objects – Window

- The window object represents the browser window that contains the DOM document.
- It is the topmost object in the DOM hierarchy and is the global scope in client-side JavaScript.
- Some common uses of the window DOM object include:
  - Accessing global variables, functions, and JavaScript objects
  - Manipulating browser window properties like innerHeight, innerWidth
  - Adding event handlers for window events like load, resize, scroll
  - Opening new browser windows and communicating between windows
  - Storing data within sessionStorage or localStorage

DOM objects – Window (cont.)

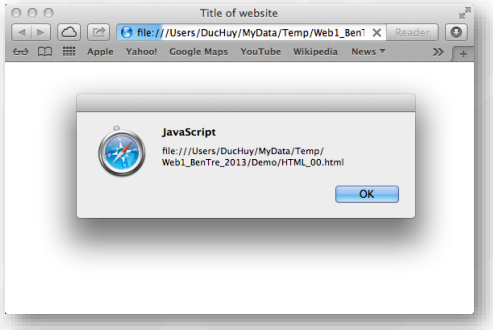
- **Properties**
  - **document**
  - event
  - history
  - location
  - name
  - navigator
  - screen
  - status

- **Methods**
  - Alert
  - Confirm
  - Prompt
  - Blur
  - close
  - Focus
  - open

DOM objects – Window (cont.)

❑ Ex. :

```
<html>
    <body>
        <script type="text/javascript">
            var curURL = window.location;
            window.alert(curURL);
        </script>
    </body>
</html>
```



DOM objects – Document

❑ The document object represents your web page.

❑ If you want to access any element in an HTML page, you always start with accessing the document object.

DOM objects – Document (cont.)

Properties

- aLinkColor
- bgColor
- body
- fgColor
- linkColor
- title
- URL
- vlinkColor
- forms[]
- images[]
- childNodes[]
- documentElement
- cookie
- ……

Methods

close
open
createTextNode(“ text ")
createElement(“HTMLtag")
getElementById(“id”)
…

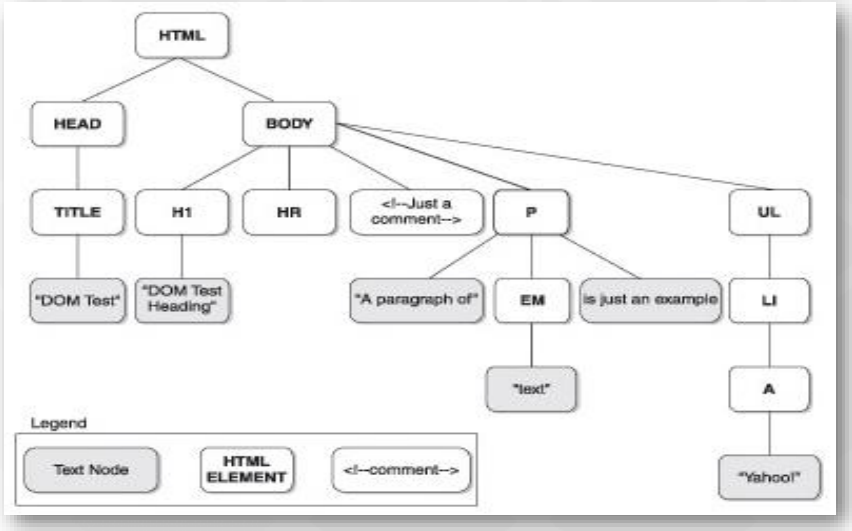
DOM objects – Document (cont.)

❑ HTML document

```
<html>
    <head>
        <title>DOM Test</title>
    </head>
    <body>
        <h1>DOM Test Heading</h1>
        <hr />
        <!-- Just a comment -->
        <p id=“p1” >A paragraph of <em>text</em> is just an example</p>
        <ul>
            <li>
                <a href=“http://www.yahoo.com” > Yahoo! </a>
            </li>
        </ul>
    </body>
</html>
```

DOM objects – Document (cont.)

❑ The HTML DOM Tree of Objects



DOM objects – Document (cont.)

❑ Finding HTML Objects

Property	Description
document.documentElement	Returns the <html> element
document.head	Returns the <head> element
document.body	Returns the <body> element
document.forms	Returns all <form> elements
document.anchors	Returns all <a> elements that have a name attribute
document.images	Returns all <img> elements
document.scripts	Returns all <script> elements
document.title	Returns the <title> element
document.URL	Returns the complete URL of the document

Finding HTML Element by Id

- **document.**getElementById(*id*)
  Find an element by element id

Ex. :

```
//<p id="id1">
//      some text
//</p>
```

Text Node: “some text”

```
const node = document.getElementById(“id1”);
const nodeName = node.nodeName; // p
const nodeType = node.nodeType; // 1
const nodeValue = node.nodeValue; // null
const text = node.innerText ; // some text
```

Finding HTML Elements by Tag/Class Name

- **document.**getElementsByTagName(name),
- **document.**getElementsByClassName(name)
  Find elements by tag name

Ex. :

```
//<p class=“cl1">
//      some text
//</p>
```

Text Node: “some text”

```
const nodes = document.getElementsByClassName(“cl1”);
const nodes2 = document.getElementsByTagName(“p”);
const nodeName = node[0].nodeName; // p
const text = nodes2[0].innerText ; // some text
```

Finding HTML Element(s) by CSS Selectors

- If you want to find a HTML element that match a specified CSS selector (id, class names, types, attributes, values of attributes, etc), use the querySelector() method.
- If you want to find all HTML elements use the querySelectorAll() method.

DOM objects – Document (cont.)

- Changing HTML Elements

Property / Method	Description
element.innerHTML = new html content	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.style.property = new style	Change the style of an HTML element
element.setAttribute(attribute, value)	Change the attribute value of an HTML element

```
const pElement = document.querySelector('p');
pElement.innerHTML = "<b>some</b> text";
pElement.setAttribute('style', 'font-size:36px;');
pElement.style.color = '#ff00aa';
```

some text

DOM objects – Document (cont.)

❑ Adding and Deleting Elements

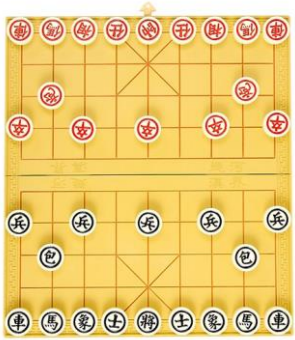
Method	Description
document.createElement(element)	Create an HTML element
document.removeChild(element)	Remove an HTML element
document.appendChild(element)	Add an HTML element
document.replaceChild(new, old)	Replace an HTML element
document.write(text)	Write into the HTML output stream

```
const newP = document.createElement('p');
const contentText = document.createTextNode('some text');
newP.appendChild(contentText);
document.body.appendChild(newP);          // some text
newP.innerText = "another text";          // another text
document.write('<h1>New content</h1>'); // New content
```

Contents

Assignment 1

- Create an HTML page that is a chessboard (either Chess or Chinese chess), allowing players to use the mouse to move the pieces (only allowing them to change positions to the correct squares).



Assignment 2

- Create a website that allows for calculations with two operands, with data validation and an interface suggestion as follows:

Bé tập tính

Số thứ nhất [ ] ○ Cộng ○ Trừ

Số thứ hai [ ] ○ Nhân ◉ Chia

Kết quả [ ] Tính

Thông báo Giá trị nhập ở ô *Số thứ nhất* không phải là số