CÔNG NGHỆ WEB

NGÔN NGỮ KỊCH BẢN PHÍA CLIENT

NỘI DUNG

- Javascript
- jQuery
- Ajax
- JSon
- Thảo luận: Thiết kế web

Javascript

- Giới thiệu:
  - JavaScript là ngôn ngữ dưới dạng script.
  - JavaScript là ngôn ngữ dựa trên đối tượng, có nghĩa là bao gồm nhiều kiểu đối tượng.
  - JavaScript có thể đáp ứng các sự kiện như tải hay loại bỏ các form. Khả năng này cho phép JavaScript trở thành một ngôn ngữ script động.
  - Javasript có thể làm việc khác nhau trên các trình duyệt khác nhau

- Nhúng JavaScript vào một file HTML theo một trong các cách sau đây:
  - Sử dụng các câu lệnh và các hàm trong cặp thẻ <SCRIPT>

```
<script type="text/javascript" language="javascript" >
<!--
    // <Các câu lệnh> ;
 -->
</script>
```

  - Sử dụng các file nguồn JavaScript

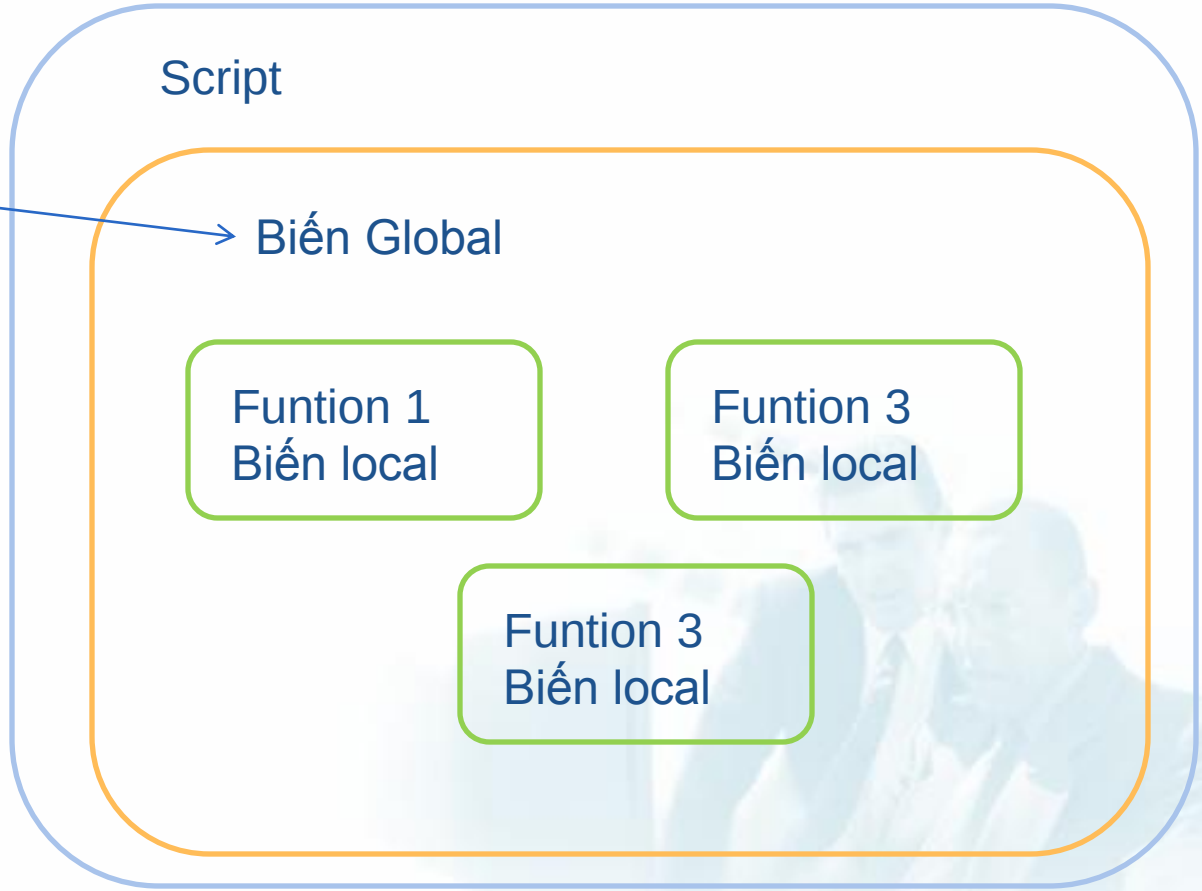
```
<script type="text/javascript" src="filename.js" >
…
</script>
```

- Nhúng JavaScript vào một file HTML theo một trong các cách sau đây:
  - Sử dụng một biểu thức JavaScript làm giá trị của một thuộc tính HTML:
  - Sử dụng thẻ sự kiện (event handlers) trong một thẻ HTML nào đó
- Khai báo tên biến:

  var tenbien [=<giá trị>];

- Phạm vi của biến:

Biến Global có thể được truy cập bởi các hàm 1, 2, 3



- Các toán tử:
  - Phép gán (=):

Kiểu gán thông thường	*Kiểu gán rút gọn*
x = x + y	x + = y
x = x – y	x - = y
x = x * y	x * = y
x = x / y	x / = y
x = x % y	x % = y

  - Phép toán so sánh: ==(bằng), != (khác), >, >=, <, <=
  - Phép toán số học: +, -, *, /, % (chia lấy phần dư), ++, --
  - Ghép chuỗi(+)
  - Phép toán Logic: && (và), || (hoặc), ! (phủ định).

- Mảng:
  - Mảng dùng để lưu một tập hợp các biến cùng tên
  - Chỉ số mảng để phân biệt biến này với biến khác, chỉ số được bắt đầu từ 0
  - Khai báo mảng:

arrayObjectName=new Array([element0, element1, … , elementN])

```
emp = new Array(3);
emp = new Array(3,2);
```

- Thêm các phần tử mảng:

  arrayObjectName[i]=<giá trị>;

- Truy cập phần tử mảng:

  arrayObjectName[i]

- Sử dụng các phương thức của mảng:
  - Join: Kết hợp các phần tử của mảng thành mọt chuỗi
  - Pop: trà về phần tử cuối cùng của mảng
  - Push: Thêm một hoặc nhiều phần tử vào cuối mảng
  - Reverse: Đảo ngược phần tử mảng
  - Shift: Xóa phần tử đầu tiên của mảng và trả về phần tử đó
  - Sort: Sắp xếp các phần tử mảng

- Các cấu trúc lệnh:
  - Cấu trúc rẽ nhánh: if ... else:

```
if (<điều kiện>)
{
  //Các câu lệnh với điều kiện đúng
}
else
{
   //Các câu lệnh với điều kiện sai
}
```

- Các cấu trúc lệnh:
  - Cấu trúc lựa chọn switch ... Case:

```
switch ( <biến> )
{     case value1:
            statements1;
            break;
            ...
      case value n-1:
            statements(n-1);
            break;
      default:
            statements(n);
}
```

- Các cấu trúc lệnh:
  - Vòng lặp for: Lặp khi điều kiện đúng

```
for (initExpr; <Điều kiện> ; incrExpr)
{
   //Các lệnh thực hiện trong vòng lặp
}
```

  - Vòng lặp while

```
while (<điều kiện>)
{
   //Các lệnh thực hiện trong vòng lặp
}
```

- Các cấu trúc lệnh:
  - Lệnh Break: Dùng để thoát khỏi vòng lặp gần nhất.
  - Lệnh continue: Dùng để bắt đầu một vòng lặp mới.
  - Vòng lặp for...in: Câu lệnh này thường được sử dụng để lặp tất cả các thuộc tính (properties) của một đối tượng.

```
for (<variable> in <object>)
{
  //Các câu lệnh
}
```

  - Lệnh new: Để tạo ra một thể hiện mới của một đối tượng

```
objectvar = new object_type( param1
  [,param2]... [,paramN])
```

- Hàm: Trong JavaScript ta dễ dàng nhận thấy có 2 loại Hàm
  - Các Hàm JavaScript đã hổ trợ sẳn (Built-in Functions)
  - Ngoài ra người dùng có thể định nghĩa ra các hàm để phục vụ cho mục đích riêng (User-defined Functions).

```
function TenHam(bien_1,bien_2,...)
{  // Thân hàm …
     return value;
}
```

- Một số hàm JavaScript (Built-in Functions):
  - isNaN(var): Kiểm tra một biến có phải là số hay không?
  - parseInt(var): Chuyển một chuỗi sang số nguyên.
  - parseFloat(var): Chuyển một chuỗi sang số Float
  - eval(""): Định giá trị cho các statement hoặc expression được lưu trữ.
  - alert("…"): Dùng để gởi một thông báo cho User
  - prompt("string_a","string_b"): Dùng để tạo ra một dialog box tương tác với User, có 2 nút là OK ,CANCEL
    - string_a: ghi một nhãn lên dialog box
    - string_b:giá trị mặc định trong text box

- JavaScript Objects:
  - Objects là các biến chứa các biến
  - Các giá trị được viết theo cặp **name : value**

```
var person = {firstName:"John",
              lastName:"Doe",
              age:50,
              eyeColor:"blue"};
```

- JavaScript Objects:
  - Tạo JavaScript Object: 3 cách
    - Sử dụng từ khóa {}

```
var person = {
    firstName:"John",
    lastName:"Doe",
    age:50
   };
```

    - Sử dụng từ khóa new Object()

```
var person = new Object();
person.firstName = "John";
person.lastName = "Doe";
person.age = 50;
```

- JavaScript Objects:
  - Tạo JavaScript Object: 3 cách
    - Sử dụng từ khóa Constructor:

```
function person(first, last, age) {
    this.firstName = first;
    this.lastName = last;
    this.age = age;
  }
var myFather = new person("John", "Doe", 50);
```

- JavaScript Objects:
  - Thuộc tính của đối tượng:
    - Truy xuất thuộc tính:
      - objectName.age// person.age
      - objectName["property"]// person["age"]
      - objectName[expression]// x = "age"; person[x]
      - for (variable in object) {
        // code to be executed
        }

```
var person = { fname: "John", lname: "Doe", age: 25 };
for (x in person) {
    txt += person[x];}
```

- JavaScript Objects:
  - Thuộc tính của đối tượng:
    - Truy xuất thuộc tính:
      - objectName.age// person.age
      - objectName["property"]// person["age"]
      - objectName[expression]// x = "age"; person[x]
      - for (variable in object) {
        // code to be executed
        }

```
var person = { fname: "John", lname: "Doe", age: 25 };
for (x in person) {
    txt += person[x];}
```

- JavaScript Objects:
  - Thuộc tính của đối tượng:
    - Thêm thuộc tính mới:
      - objectName.NewProperty=Values;

```
person.nationality = "English";
```

    - Xóa thuộc tính:
      - delete objectName.Property;
      - delete objectName[“Property”]

```
delete person.age;   // or delete person["age"];
```

- JavaScript Objects:
  - Phương thức của đối tượng:
    - Tạo phương thức:
      - methodName: function() {// code lines}

```
var person = {
    firstName: "John",
    lastName: "Doe",
     fullName: function () {
        return this.firstName +
            " “ + this.lastName;
    }
}
```

- JavaScript Objects:
  - Phương thức của đối tượng:
    - Tạo phương thức:

```
function person(firstName, lastName) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.changeName = function (name) {
        this.lastName = name;
    };
}
```

- JavaScript Objects:
  - Mộ số đối tượng Built in:
    - String: Đối tượng string được dùng để thao tác và làm việc với chuỗi ký tự.
    - Đối tượng Math: được dùng để thao tác các phép toán học
    - Đối tượng Date: là đối tượng có sẵn chứa thông tin về ngày giờ.

- Mô hình đối tượng (Document Object Model - DOM):
  - Javascript là ngôn ngữ dựa trên đối tượng: Cho phép người phát triển theo module và có thể được sử dụng lại
  - Javascript không hoàn toàn là ngôn ngữ hướng đối tượng, nhưng nó hỗ trợ một vài đặc điểm của đối tượng.

- DOM:
  - Tìm kiếm các phần tử HTML:
    - document.getElementById(): Tìm phần tử theo id
    - document.getElementsByTagName(): Tìm phần tử theo thẻ
    - document.getElementsByClassName(): Tìm phần tử theo tên class

- DOM:
  - Thay đổi phần tử HTML:
    - *element*.innerHTML= Thay đổi bên trong phần tử
    - *element*.*attribute*= Thay đổi thuộc tính của một phần tử.
    - *element*.setAttribute*(attribute,value)* Thay đổi thuộc tính của phần tử
    - *element*.style.*property*= Thay đổi style của phần tử.

- DOM:
  - Thêm hoặc xóa phần tử:
    - *document.createElement():* Tạo phần tử HTML
    - document.removeChild(): Xóa HTML element
    - document.appendChild(): Thêm một HTML element
    - document.replaceChild(): Thay thế HTML element
    - document.write(*text*): Viết luồng HTML output
  - Thêm điều khiển sự kiện:
    - document.getElementById(*id*).onclick=function(){*code*}: Thêm sự liện onclick()

jQuery

- Giới thiệu:
  - jQuery là thư viện JavaScript Library, cung cấp rất nhiều frameworks về Javascript.
  - jQuery đơn giản hóa lập trình JavaScript
  - jQuery rất dễ cho việc học.
- Sử dụng jQuery:
  - Sử dụng như file JavaScrip (http://jquery.com/download/)

```
<head>
    <script src="jquery-1.11.3.min.js"> </script>
</head>
```

- Sử dụng jQuery:
  - + Sử dụng host jQuery:

```
<script
src="https://ajax.googleapis.com/ajax/libs/
jquery/1.11.3/jquery.min.js"></script>
```

Hoặc:

```
<script
src="http://ajax.aspnetcdn.com/ajax/jQuery/
jquery-1.11.3.min.js"></script>
```

- Cú pháp jQuery:
  - Cú pháp cơ bản: **$(*selector*).*action*()**

  Trong đó:
    - $ chỉ định định nghĩa/truy cập jQuery
    - *selector* truy vấn hoặc tìm các phần tử HTML elements
    - jQuery *action*() hành động thực hiện trên các phần tử
  - Ví dụ:

```
$(this).hide() - hides the current element.
$("p").hide() - hides all <p> elements.
$(".test").hide() - hides all elements with class="test".
$("#test").hide() - hides the element with id="test".
```

- Document Ready Event: Dùng để ngăn cản code jQuery chạy trước khi document kết thúc việc loading.

```
$(document).ready(function(){
    // jQuery methods go here...
});
```

Hoặc

```
$(function(){
// jQuery methods go here...
});
```

- jQuery Selectors

Ví dụ	Ý nghĩa
$("p")	Chọn tất cả các thẻ <p> (element Selector)
$("#test")	Chọn thẻ có id="test“ (#id Selector)
$(".test")	Chọn các thẻ có thuộc tính class="test“ (.class Selector)
$("*")	Chon tất cả các phần tử
$(this)	Chọn thẻ HTML hiện tại
$("p.intro")	Chọn tất cả thẻ <p> có thuộc tính class="intro"
$("p:first")	Chọn thẻ <p> đầu tiên
$("ul li:first")	Chọn thẻ đầu tiên của thẻ đầu tiên
$("ul li:first-child")	Chọn thẻ đầu tiên của mọi thẻ

- jQuery Selectors

Ví dụ	Ý nghĩa
$("[href]")	Chọn tất cả thẻ có thuộc tính href
$("a[target='_blank']")	Chọn tất cả thẻ <a>có thuộc tính target="_blank"
$("a[target!='_blank']")	Chọn tất cả thẻ <a>có thuộc tính target KHÔNG ="_blank"
$(":button")	Chọn tất cả thẻ <button> và các thẻ <input> kiểu type="button"
$("tr:even")	Chon tất cả các thẻ <tr> chẵn.
$("tr:odd")	Chọn tất cả các thẻ <tr> lẻ.

Ví dụ:

- jQuery Event Methods: Dùng để đáp ứng các sự kiện trên các phần tử HTML
  - Cú pháp:

```
$(selector).action(
function(){
    // action goes here!! });
```

  - Ví dụ:

```
$(document).ready(function () {
    $("input").focus(function (){
        $(this).css("background-color", "#ccdddd");});
    $("input").blur(function () {
        $(this).css("background-color", "#ffffff");});
});
```

- jQuery Event Methods:
  - Một số sự kiện hay dùng:
    - click()
    - dblclick()
    - mouseenter()
    - mouseleave()
    - hover()
    - focus()
    - blur()

- Một số hiệu ứng của jQuery:
  - hide() and show():
    - $(*selector*).hide(*speed,callback*) dùng để ẩn và
    - $(*selector*).show(*speed,callback*) hiện các phần tử HTML;
    - $(*selector*).toggle(*speed,callback*) kết hợp hide() và show()
    - Ví dụ:

```
$(document).ready(function () {
        $("#hide").click(function () {
            $("p").hide(); });
 });
```

- Một số hiệu ứng của jQuery:
  - Fading: jQuery có thể làm phai nhạt phần tử trong và ngoài tầm nhìn.
    - $(selector).fadeIn(speed,callback);
    - $(*selector*).fadeOut(*speed,callback*);
    - $(*selector*).fadeToggle(*speed,callback*);
    - $(*selector*).fadeTo(*speed,opacity,callback*); Làm mờ đến một độ mờ nhất định *opacity* nhận giá trị trong khoảng 0-1
    - Ví dụ:

- Một số hiệu ứng của jQuery:
  - Fading:
    - Ví dụ:

```
$("button").click(function () {
    $("#div1").fadeTo("slow", 0.15);
    $("#div2").fadeTo("slow", 0.4);
    $("#div3").fadeTo("slow", 0.7);
});
```

- Một số hiệu ứng của jQuery:
  - Sliding: Trượt các phần tử.
    - $(*selector*).slideDown(*speed,callback*);
    - $(*selector*).slideUp(*speed,callback*);
    - $(*selector*).slideToggle(*speed,callback*);
    - Ví dụ:

```
$(document).ready(function () {
    $("#flip").click(function () {
        $("#panel").slideToggle("slow");
    });
});
```

- Một số hiệu ứng của jQuery:
  - Animation:
    - $(*selector*).animate({*params*},*speed*,*callback*);
    - Ví dụ:

```
$("button").click(function () {
        $("div").animate({
            left: '250px',
            height: '+=150px',
            width: '+=150px'
        });
    });
```

- Một số hiệu ứng của jQuery:
  - Stop Animations: Dừng quá trình sliding hoặc animations
    - $(*selector*).stop(*stopAll*,*goToEnd*);
    - Ví dụ:

```
$("#flip").click(function () {
        $("#panel").slideDown(5000);
  });
  $("#stop").click(function () {
        $("#panel").stop();
  });
```

- Một số hiệu ứng của jQuery:
  - Chaining: Ta có thể kết chuỗi actions/methods:
    - Ví dụ:

```
$("button").click(function(){
  $("#p1").css("color", "red")
      .slideUp(2000)
      .slideDown(2000);
});
```

- jQuery với HTML:
  - text() - thiết lập hoặc trả về về nội dung text của các phần tử được lựa chọn.
  - html() - thiết lập hoặc trả về về nội dung của các phần tử được lựa chọn (bao gồm HTML markup)
  - val() - thiết lập hoặc trả về giá trị của form fields
  - attr() – phương thức dùng để thiết lập hoặc lấy giá trị của thuộc tính.
    - Ví dụ:

```
$("button").click(function(){
    $("#w3s").attr("href", "http://www.w3schools.com/jquery");
});
```

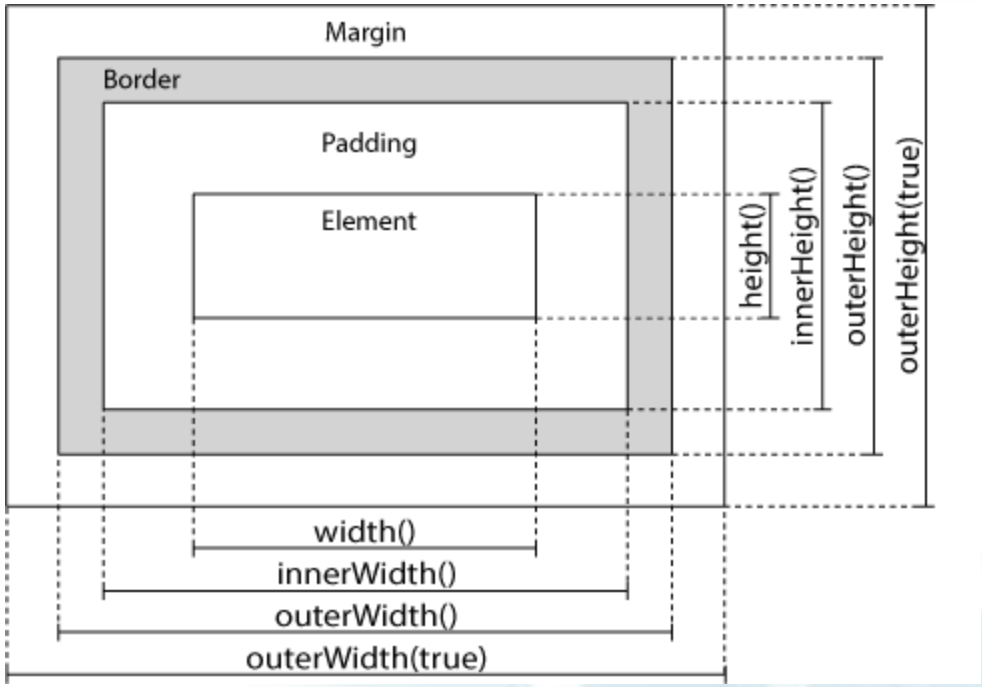
- jQuery với HTML:
  - append() - Chèn nội dung vào cuối phần tử đã chọn
  - prepend() – Chèn nội dung vào đầu phần tử đã chọn.
  - after() - Chèn nội dung sau phần tử đã chọn.
  - before() – Chèn nội dung trước phần tử đã chọn
  - remove() – Xóa phần tử đã chọn (bao gồm các phần tử con của nó)
  - empty() – Xóa các phần tử con của phần tử đã chọn.

- jQuery với HTML:
  - Get and Set CSS Classes
    - addClass() – Thêm một hoặc nhiều class
    - removeClass() – Xóa một hoặc nhiều class
    - toggleClass() - Toggles giữa adding/removing
    - css("*propertyname*","*value*") - Thiết lập hoặc trả về style attribute
    - Ví dụ:

- jQuery với HTML:
  - Get and Set CSS Classes
    - Ví dụ:

```
$("button").click(function(){
    $("h1, h2, p").addClass("blue");
    $("div").addClass("important");
});
//
$("p").css({"background-color": "yellow", "font-size":
"200%"});
```

- jQuery với HTML:
  - Dimensions
    - width()
    - height()
    - innerWidth()
    - innerHeight()
    - outerWidth()
    - outerHeight()



- jQuery với HTML:
  - Dimensions
    - Ví dụ:

```
$("button").click(function(){
    var txt = "";
    txt += "Outer width: " + $("#div1").outerWidth() + "</br>";
    txt += "Outer height: " + $("#div1").outerHeight();
    $("#div1").html(txt);
});
$(document).ready(function(){
    $("button").click(function(){
        $("#div1").width(500).height(500);
    });
});
```

JSON

- JSON (**J**ava**S**cript **O**bject **N**otation) là định dạng cho việc lưu trữ và truyền dữ liệu với dung lượng nhỏ
- JSON thường được sử dụng khi dữ liệu truyền từ server đến web page.
- Định dạng của JSON tương tự như định dạng các đối tượng Javascript

JSON

- Cú pháp:
  - Data lưu trữ trong cặp name/value
  - Data được phân cách bởi dấu phảy ‘,’
  - Các đối tượng được bao bởi cặp {}

```
{"firstName":"John", "lastName":"Doe"}
```

  - Mảng được bao bởi cặp ngoặc vuông []

```
"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]
```

- Chuyển từ JSON Text sang JavaScript Object: Sử dụng JSON.parse() để chuyển chuỗi string sang JavaScript object

```
<script>
    // Chuỗi có định dạng JSON
    var text = '{"employees":[' +
    '{"firstName":"John","lastName":"Doe" },' +
    '{"firstName":"Anna","lastName":"Smith" },' +
    '{"firstName":"Peter","lastName":"Jones" }]}';
    //Chuyển chuỗi sang JavaScript object
    obj = JSON.parse(text);
    document.getElementById("demo").innerHTML =
    obj.employees[1].firstName + " " + obj.employees[1].lastName;
</script>
```

AJAX

- AJAX là công nghệ trao đổi dữ liệu với máy chủ, và cập nhật các phần của một trang web mà không cần tải lại toàn bộ trang.
- AJAX = Asynchronous JavaScript and XML.

- Một số phương thức của Ajax.
  - ajax() method: Được sử dụng để thực hiện một AJAX (asynchronous HTTP) request.
    - Cú pháp: `$.ajax({name:value,name:value,...})` với một số cặp names/values:
      - url: đường dẫn đến file lấy kết quả
      - type: post hoặc get
      - dataType: kiểu dữ liệu trả về, có thể là json, xml, script hoặc text
      - data: Chỉ định dữ liệu được gửi đến Server.
      - success : là hàm xử lý kết quả trả về, nó có tham số lưu trữ kết quả trả về.

- Một số phương thức của Ajax.
  - ajax() method:
    - Ví dụ 1:

```
$('.btn-delete').off('click').on('click',function(e){
      e.preventDefault();
      $.ajax({
          data: { id: $(this).data('id') },
          url: '/Cart/Delete',
          dataType: 'json',
          type: 'POST',
          success: function (res) {
              if (res.status == true) {
                  window.location.href = "/gio-hang";
               }
           }})
    });
```

- Một số phương thức của Ajax.
  - ajax() method:
    - Ví dụ 2:

```
$(document).ready(function () {
        $("button").click(function () {
            $.ajax({
                url: "data/TextFile.txt",
                type: "GET",
                success: function (result) {
                 $("#Test").html(result);
                }
            });
        });
    });
```

- Một số phương thức của Ajax.
  - ajax() method:
    - Ví dụ 3:

```
$(document).ready(function () {
    $("button").click(function () {
     $.ajax({url: "data/TextFile.txt", type: "GET",
              success: function (result) {
                var obj = JSON.parse(result);
                var out = "<table>"; var i;
                for (i = 0; i < obj.employees.length; i++) {
                       out += '<tr> <td>' +
                       obj.employees[i].firstName + '</td><td>' +
                       obj.employees[i].lastName + '</td></tr>'; }
             $("#Test").html(out);}
           });
       });
   });
```

- Một số phương thức của Ajax.
  - load() Method: Được sử dụng để loads dữ liệu từ server và chèn dữ liệu vào element đang chọn.
    - Cú pháp:
      `$(selector).load(url,data,function(response,status,xhr))` với
      - url: chỉ định URL muốn load.
      - data: Chỉ định dữ liệu gửi đến sever cùng với request.
      - responseTxt – Chứa nội dung trả về nếu load thành công.
      - statusTxt – Chứa trạng thái khi gọi hàm
      - xhr – chứa đối XMLHttpRequest

- Một số phương thức của Ajax.
  - load() Method:
    - Ví dụ:

```
$(document).ready(function () {
     $("button").click(function () {
       $("#div1").load("data/TextFile3.txt #p1", function
(responseTxt, statusTxt, xhr) {
           if (statusTxt == "success")
               alert("External content loaded successfully!");
           if (statusTxt == "error")
           alert("Error:"+ xhr.status + ": " + xhr.statusText);
            });
        });
   });
```

- Một số phương thức của Ajax.
  - post() Method: Được sử dụng để loads dữ liệu từ server sử dụng HTTP POST request..
    - Cú pháp:

```
$(selector).post(URL,data,function(data,status,xhr),dataType)
```

  - url: chỉ định URL muốn load.
  - data: Chỉ định dữ liệu gửi đến sever cùng với request.
  - function(data,status,xhr): Chỉ định hàm chạy khi request thành công.
    - data – Chứa dữ liệu trả về từ request.
    - status – Chứa trạng thái request "success", "notmodified", "error", "timeout", or "parsererror"
    - xhr – chứa đối XMLHttpRequest
  - dataType: "xml“, "html“, "text“, "script“, "json"

- Một số phương thức của Ajax.
  - post() Method:
    - Tương đương:

```
$.ajax({    type: "POST",
            url: url,
            data: data,
            success: success,
            dataType: dataType
        });
```

- Một số phương thức của Ajax.
  - post() Method:
    - Ví dụ:

```
$(document).ready(function () {
    $("input").keyup(function () {
        var txt = $("input").val();
        $.post("Default.aspx",{sendData:txt},function (result){
                $("span").html(result);
            }); });
    });
// Xử lý trên trang Default.aspx:
string getResult = Request.Form["sendData"].ToString();
getResult = "Dữ liệu đã gửi " + getResult;
Response.Write(getResult);
```

- Một số phương thức của Ajax.
  - get() Method: Được sử dụng để loads dữ liệu từ server sử dụng HTTP GET request.
    - Cú pháp:

```
$(selector).get(URL,data,function(data,status,xhr),dataType)
```

  - url: chỉ định URL muốn load.
  - data: Chỉ định dữ liệu gửi đến sever cùng với request.
  - function(data,status,xhr): Chỉ định hàm chạy khi request thành công.
    - data – Chứa dữ liệu trả về từ request.
    - status – Chứa trạng thái request "success", "notmodified", "error", "timeout", or "parsererror"
    - xhr – chứa đối XMLHttpRequest
  - dataType: "xml“, "html“, "text“, "script“, "json"

- Một số phương thức của Ajax.
  - get() Method :
    - Tương đương:

```
$.ajax({
  url: url,
  data: data,
  success: success,
  dataType: dataType
});
```

- Một số phương thức của Ajax.
  - get() Method :
    - Ví dụ:

```
$(document).ready(function () {
    $("input").keyup(function () {
        var txt = $("input").val();
        $.get("Default.aspx",{sendData:txt},function (result){
                $("span").html(result);
            }); });
    });
// Xử lý trên trang Default.aspx:
string getResult = Request.QueryString["sendData"].ToString();
getResult = "Dữ liệu đã gửi " + getResult;
Response.Write(getResult);
```

- Một số phương thức của Ajax.
  - getJSON(): Được sử dụng để nhận JSON data sử dụng AJAX HTTP GET request..
    - Cú pháp:

```
$(selector).getJSON(URL,data,function(data,status,xhr))
```

    - url: chỉ định URL muốn load.
    - data: Chỉ định dữ liệu gửi đến sever cùng với request.
    - function(data,status,xhr): Chỉ định hàm chạy khi request thành công.
      - data – Chứa dữ liệu trả về từ request.
      - status – Chứa trạng thái request "success", "notmodified", "error", "timeout", or "parsererror"
      - xhr – chứa đối XMLHttpRequest

- Một số phương thức của Ajax.
  - getJSON():
    - Tương đương:

```
$.ajax({
  dataType: "json",
  url: url,
  data: data,
  success: success
});
```

- Một số phương thức của Ajax.
  - getJSON():
    - Ví dụ 1:

```
$(document).ready(function () {
    $("button").click(function () {
    $.getJSON("data/data.js", function (result){
        $.each(result, function (i, field) {
            $("div").append(field + " ");
        });});
    });
});
```

- Một số phương thức của Ajax.
  - getJSON():
    - Ví dụ 2:

```
$(document).ready(function () {
   $("button").click(function () {
      $.getJSON("data/dataArray.json",function(result){
          $.each(result, function(idx, obj) {
              $.each(obj, function (key, value) {
              $("div").append(key + ": " + value);
                  console.log(key + ": " + value);
              });});
      });
   });
});
```

- Một số phương thức của Ajax.
  - serialize() Method: Tạo ra chuỗi mã hóa URL bằng cách chuyển đổi form values.
    - Các giá trị serialized values có thể được sử dụng trong URL query string khi tạo ra AJAX request
    - Cú pháp:

```
$(selector).serialize()
```

    - Ví dụ:

```
$(document).ready(function () {
    $("button").click(function () {
        $("div").text($("form").serialize());
    });
});
```

- Một số phương thức của Ajax.
  - serializeArray() Method: Mã hóa tập các phần tử form như mọt mảng gồm các cặp name/value.
    - Tạo ra một mảng các đối tượng JavaScript, sẵn sàng cho mã hóa như JSON string
    - Cú pháp:

```
$(selector).serializeArray()
```

    - Ví dụ:

```
$(document).ready(function () {
    $("button").click(function () {
        var x = $("form").serializeArray();
        $.each(x, function (i, field) {
            $("#results").append(field.name+":"+field.value+);
        });
    });
});
```

THẢO LUẬN – CÂU HỎI

THẢO LUẬN

- Thảo luận quy trình thiết kế ứng dụng web
- Các hiệu ứng Javascript và Jquery
- JSON
- AJAX

BÀI TẬP

- Thực hành các ví dụ
- Tìm hiểu mở rộng về BOOTSTRAP.
- Thiết kế website cho đề tài được giao.