DEV WEB

Front-end

HTML

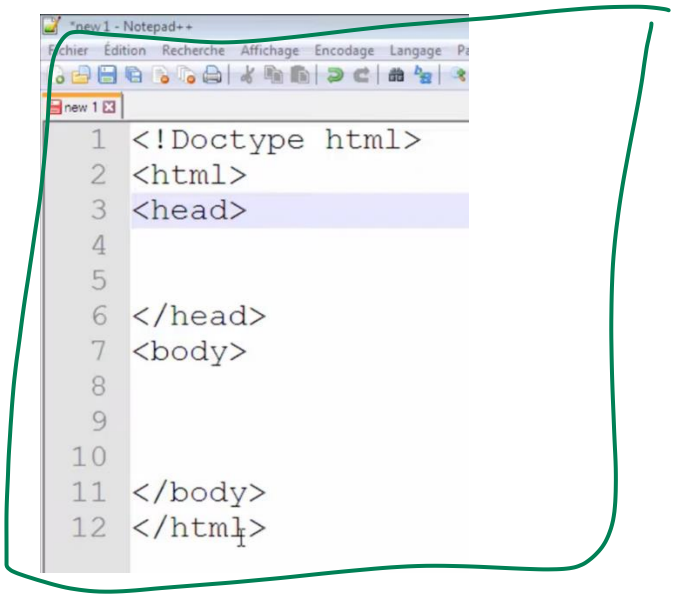
Hypertext Markup Langage

Un élément : est constitué d'une balise ouvrante, une balise fermente et le contenu entre ces deux balises.

Un attribut : appartient à l'élément et se trouve dans la balise ouvrante

exple : <h1 style = ''color : red''>----------</h1>

Page vierge→



Basics

<hn>----------</hn>

n de 1 à 6 pour l'écriture d'un titre 1 la taille la plus >, 6 la plus <

</hn>

Pour l'écriture d'un paragraphe

 Saut de ligne (élément vide)

  Pour l'espace (s'écrit entre deux mots)

< !-- Commentaire -- >

<h1 style = ''color : red''>----------</h1> Modification de la couleur du texte entre <h1>----------</h1>

<body style = ''background-color : green''>----------</body>

Modification de la couleur de la page (background)

La balise style

< balise style = ''propriété: valeur''>

Marqueurs de style (mds) :

------<mds>---------- </mds>-------

Balise de style	Effet Visuel
<STRONG> …</STRONG> ou <B>...</B>	**Gras (Bold)**
<I> … </I>	*Italique*
<U>...</U>	Souligné
<FONT SIZE=?>...</FONT>	Taille de caractère (Font size)
<FONT COLOR="#$$$$$$"> </FONT>	Couleur de caractère (Font color)
<!-- *** -->	Commentaires
<BIG> et </BIG>	Police plus grande
<SMALL> et </SMALL>	Police plus petite
<CENTER></CENTER>	Centrage (Center)

FONT SIZE = 300% **(la taille du texte*3)**

Balise de style	Effet Visuel
<PRE>...</PRE>.	texte préformaté
<Q> et </Q>	Encadre le texte par des guillemets
_{et}	Texte en $_{\text{Indice}}$
^{et}	Texte en ^{Exposant}
<ABBREV> et </ABBREV>	Abréviation
<ACRONYM> et </ACRONYM>	Acronyme
<note> et </note>	Pour écrire une note
<fn> et </fn>	Permet d'avoir une note de fin de page
<ADDRESS>...</ADDRESS>	pour indiquer une adresse
<AU> et </AU>	L'auteur
<CITE> et </CITE>	*Citation*

<mark>----------</mark> **to highlight a text**

~~----------~~ **barre le texte**

Les liens

Transformer un texte en lien :

<a href='' *on insert notre lien* **''>** **le texte qui s'affiche sous forme de lien** **<a>**

<a href='' *on insert notre lien* **'' target= ''_blank '' >** **le texte qui s'affiche sous forme de lien** **<a>**

➔ **Ouvrir le lien dans un autre onglet**

Passer vers une partie de la même page en cliquant sur un lien :

<a href='' #nom_id ''> **le texte qui s'affiche sous forme de lien** **<a>**

…….

<h2 id = ''nom_id''>----------</h2>

Ouvrir une autre page html :

<a href=’’ chemin de la page / nom de la page.html ‘’> le texte qui s’affiche sous forme de lien <a>

(Chemin de la page si elle se trouve dans un dossier ou sous dossier sinon le nom.html est suffisant)

Afficher un message lorsque la souris passe sur le lien :

<a href=’’ *on insert notre lien’’* **text= ‘‘le message ‘’> le texte qui s’affiche sous forme de lien <a>**

Les images

<img src= ‘‘nom de l’image.extension’’ alt = ‘’texte alternatif pour l’image (on peut le laisser vide)’’ >

Une image cliquable

<a href=’’ *on insert notre lien* **‘’> <img src= ‘’ nom.ext’’ alt= ‘’ txt’’> <a>**

Modifier les dimensions d’une image

<img src= ‘’ nom.ext’’ alt= ‘’ txt’’ height = ‘’300’’>

300→ 300 pixels

Les listes

Conteneur	Type de liste	Effet Visuel
  - article 1 - article 2 	Ordonnée	1. article1 2. article2 3. article3
  - article 1 - article 2 	Non ordonnée	• article1 • article2 • article3
<dl> <dt>Terme</dt> <dd>Définition</dd> </dl>	De définition	article 1 définition 1 article 2 définition 2

Les tableaux

```
<table border = ''2''>
    <tr>
        <th> Avec un ballon </th>
        <th> Sans ballon</th>
    </tr>
    <tr>
        <td> Football </td>
        <td> Box</td>
    </tr>
    <tr>
        <td> HandBall </td>
        <td> Natation</td>
    </tr>
    <tr>
        <td> Basketball </td>
        <td> karaté</td>
    </tr>
</table>
```

th = table heather

tr = table row

td = table data

output :

avec un ballon	sans ballon
FootBall	Box
HandBall	Natation
BasketBall	karaté

colspan

<th colespan = ‘’nbr de colonne’’> Sport </th>

Expl colespan = ‘’2’’

Sport	
avec un ballon	sans ballon
FootBall	Box
Natation	
BasketBall	karaté

Rowspan : le mme principe sur les lignes

Les métadonnées

Bash le site ytl3 f le moteur de recherche

<head>

<meta name=’’description’’ content=’’ ce site sert à ……’’

<meta name=’’keywords’’ content=’’ html,css,devweb,….’’

</head>

Une ligne horizontale

<hr> : toute une ligne

<hr width =’’50%’’> : moitié d’une ligne

<hr width =’’200px’’ align = ‘’center’’> : 200 pixels , center/left/right

Un texte préformaté

Bash le site y affichi un texte tel qu'il est kndiroh mabin

<pre>

</pre>

Exple :

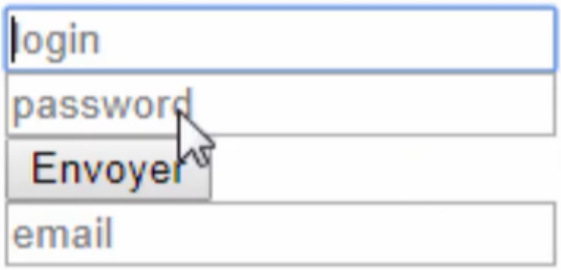
```
<pre>
                                  (####)
                                 (#######)
                                (#########)
                               (#########)
                              (#########)
                             (#########)
                            (#########)
                           (#########)
                          (########)
                         (#########)
   _____               (#########)     |\/\/\/|     /\ /\  /\                /\
  /     \              (#########)     |      |     | v  \/  \---.     .----/  \----.
 \/\/     |             (#########)     |      |      \_        /       \          /
  |  (o) (o)             (o) (o) (##)     |      |       \        /         \        /
  C    .---_)          ,_C      (##)     |  (o) (o)       (o) (o)  <__.    .--\ (o) (o) /__.
   | |.___|          /____,    (##)     C        _)     _C          /      \     ()     /
   |  \__/             \      (#)        | ,___|      /____,    )  \       >   (C_)   <
   /_____\              |     |           |   /          \      /----'     /___\____/___\
  /_____/ \            OOOOOO            /____\            OOOOO              /|     |\
 /         \           /      \          /      \          /     \             /       \
   HOMER               MARGE             BART              LISA                MAGGIE
</pre>
```

Caractères spéciaux

Caractère	Code HTML	Caractère	Code HTML	Caractère	Code HTML	Caractère	Code HTML	Caractère	Code HTML
Á	Á	©	©	Î	Î	œ	œ	³	³
á	á	¤	¤	î	î	Ò	Ò	ß	ß
Â	Â	°	°	¡	¡	ò	ò	Þ	þ
â	â	÷	÷	Ì	Ì	ª	ª	þ	þ
´	´	É	É	ì	ì	º	º	×	×
Æ	&Aelig	é	é	¿	¿	Ø	Ø	Ú	Ú
æ	æ	Ê	Ê	Ï	Ï	ø	ø	ú	ú
À	À	ê	ê	ï	ï	Õ	Õ	Û	Û
à	à	È	È	«	«	õ	õ	û	û
&	&	è	è	<	<	Ö	Ö	Ù	Ù
Å	Å	Ð	ð	¯	&masr;	ö	ö	ù	ù
å	å	ð	ð	µ	µ	¶	¶	¨	¨
Ã	Ã	Ë	Ë	·	·	±	±	Ü	Ü
ã	ã	ë	ë	Space		£	£	ü	ü
Ä	Ä	€	€	¬	¬	"	"	Ý	Ý
ä	ä	½	½	Ñ	Ñ	»	»	ý	ý
¦	¦	¼	¼	ñ	ñ	®	®	¥	&yen
Ç	Ç	¾	¾	Ó	Ó	§	§	Y	Ÿ
ç	ç	>	>	ó	ó			ÿ	ÿ
¸	¸	Í	Í	Ô	Ô	¹	¹		
¢	¢	í	í	ô	ô	²	²		

Le formulaire

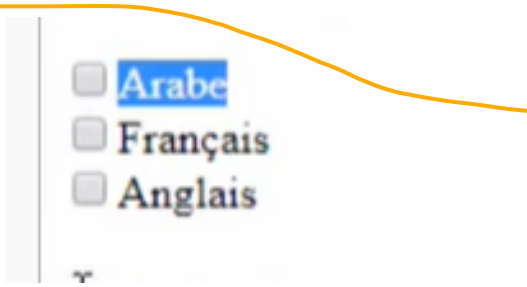
```
<form>
    <input type=''text'' size=''20'' placeholder= ''login'' ><br>
    <input type=''password'' maxlength =''20''  placeholder=
''password''><br>
    <input type=''submit'' value = ''envoyer''><br>
    <input type=''email'' placeholder= ''email''><br>
</form>
```




```
<button><img src= ''image.extension''>Valider </button>
```

Checkbox

```
<input type=''checkbox'' >Arabe<br>
<input type=''checkbox'' >Français<br>
<input type=''checkbox'' >Anglais
```



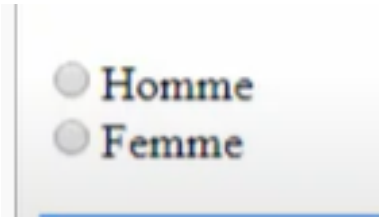

```
<input type=''checkbox''checked >Arabe<br>
```

Si on veut que Arabe soit checked par defaut

RadioButton

<input type=’’radio’’ name=’’abc ‘’ value=’’homme ‘’ >Homme

<input type=’’radio’’ name=’’abc‘’ value=’’femme >Femme



name=’’un_nom_qlq ‘’ pour sélectionner qu’un seul bouton

value= ‘’nom’’ si on veut dynamiser la page(avec php ou js..) on a besoin du nom de cette valeur envoyée.

Text area

<textarea >rows=10 cols=20<textarea>

<input type=’’number’’ > : zone de texte où ne peut saisir que des nbrs

<input type=’’url’’ >

<input type=’’tel’’ >

Date

<input type=’’date’’ >

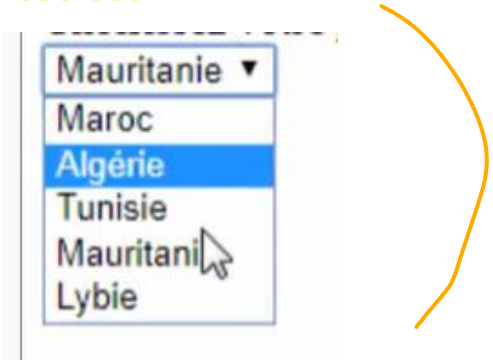
Heure

<input type=’’time’’ >

Liste déroulante

```
<select  name=’’pays’’>
    <option value=’’Maroc ‘’ >Maroc</option>
    <option value=’’ Algerie’’ >Algerie</option>
    <option value=’’ Tunisie ‘’> Tunisie </option>
    <option value=’’ Mauritanie ‘’>Mauritanie</option>
    <option value=’’Lybie ‘’> Lybie</option>
```

</select >



<fieldset>

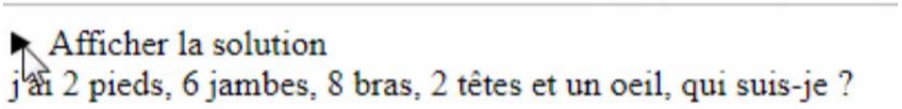
<legend>Informations personnelles</legend>

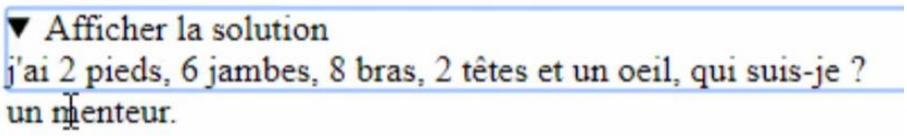
……..

</fieldset>

fieldset: tadir 2itar ela dkshi li lwst


```
<body>
<details>
<summary>Afficher la solution<br>
j'ai 2 pieds, 6 jambes, 8 bras, 2 têtes et un oeil, qui suis-je
</summary>
un menteur.
</details>
</body>
```





https://www.w3schools.com/tags/ref_attributes.asp

CSS

Cascading Style Sheets

Un sélecteur : h1

Une propriété: backgroung-color

```
<head>
<style>
 h1{
        backgroung-color: green;
}
</style>
</head>
<body>
<h1>UN TITRE</h1>
</body>
```

Best way to link CSS with HTML:

```
<head>
<link rel=’’ style sheet’’ href=’’ le chemin du document CSS’’>
</head>
```

Comments : /* */

Selecteur multiple

```
h1,h2,p{
        backgroung-color: green;
```

}

Selecteur universel

```
*{
        backgroung-color: green;
}
```

Kitebe9 ela ga3 les elts

Class (.) : bsh ntbqo un ppté ela pls elts (bzf d les h1 pr exple)

Id (#) : bsg tbq ela h1 wa7d mn les h1

La balise Span

Doc HTML

```
<p> mon premier paragraphe <span>dans ma page principale</span></p>
```

Doc CSS

```
span{
    color:red;
    background-color:green;
}
```

Output :

mon premier paragraphe dans ma page principale

La balise DIV

Doc HTML

```
<div>
<p> mon deuxième paragraphe</p>
<p> mon troisième paragraphe</p>
</div>
```

Doc CSS

```
div{
    font-family:Verdana;
}
```

Le texte

Font-size : npx ; **n un nbr**

Font-weight : x ; **x : bold / normal / lighter / bolder**

Color: rgba(, , ,x);

x de 0 à 1 : opacité du texte

Color: #FF8500;

text-align : x ; **x : center/ left / right**

text-align-last **La fin du texte**

text-transform : x ; **x : capitalize/uppercase/lowercase**

text-indent : npx **bsh nk7zo le texte**

Property	Description
text-decoration	Sets all the text-decoration properties in one declaration
text-decoration-color	Specifies the color of the text-decoration
text-decoration-line	Specifies the kind of text decoration to be used (underline, overline, etc.)
text-decoration-style	Specifies the style of the text decoration (solid, dotted, etc.)
text-decoration-thickness	Specifies the thickness of the text decoration line

letter-spacing	Specifies the space between characters in a text
line-height	Specifies the line height
text-indent	Specifies the indentation of the first line in a text-block
white-space	Specifies how to handle white-space inside an element
word-spacing	Specifies the space between words in a text

Text-shadow : a , b ,c, color ;

a : bsh7l ayk7z le texte horizontalement

b : bsh7l ayk7z le texte verticalement

c : dababia (blurr)

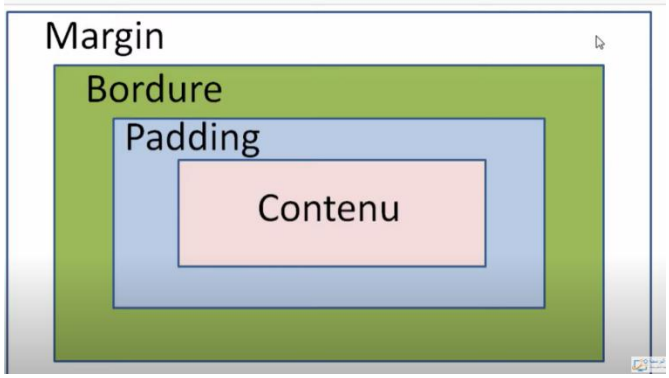
border : 6px solid #color ;

solid/dashed/doubled/dotted

pour des effets 3d : groove/ridge/inset/outset

border-x : x :bottom/top/left/right

Le modèle des boites :




```
div{
    background-color:green;
    padding:20px;
    border:10px solid blue;
    margin:30px;
    box-shadow:-2px -2px 10px orange;
}
```

Les liens :

```
a:link{
    color:blue;
    text-decoration:none;
}
a:visited{
    color:gray;
}
a:hover{
    color:red;
    text-decoration:underline;
    background-color:yellow;
    font-weight:bold;
}
a:active{
    color:orange
}
```

Les tables

```
<!DOCTYPE html>
<html>
<head>
<style>
#customers {
  font-family: Arial, Helvetica, sans-serif;
  border-collapse: collapse;
  width: 100%;
}

#customers td, #customers th {
  border: 1px solid #ddd;
  padding: 8px;
}

#customers tr:nth-child(even){background-color: #f2f2f2;}

#customers tr:hover {background-color: #ddd;}

#customers th {
  padding-top: 12px;
  padding-bottom: 12px;
  text-align: left;
  background-color: #04AA6D;
  color: white;
}
</style>
</head>
<body>
```

```
<h1>A Fancy Table</h1>

<table id="customers">
  <tr>
    <th>Company</th>
    <th>Contact</th>
    <th>Country</th>
  </tr>
  <tr>
    <td>Alfreds Futterkiste</td>
    <td>Maria Anders</td>
    <td>Germany</td>
  </tr>
  <tr>
    <td>Berglunds snabbköp</td>
    <td>Christina Berglund</td>
    <td>Sweden</td>
  </tr>
  <tr>
    <td>Centro comercial Moctezuma</td>
    <td>Francisco Chang</td>
    <td>Mexico</td>
  </tr>
  <tr>
    <td>Ernst Handel</td>
    <td>Roland Mendel</td>
    <td>Austria</td>
  </tr>
  <tr>
    <td>Island Trading</td>
```

```
  </tr>
  <tr>
    <td>Königlich Essen</td>
    <td>Philip Cramer</td>
    <td>Germany</td>
  </tr>
  <tr>
    <td>Laughing Bacchus Winecellars</td>
    <td>Yoshi Tannamuri</td>
    <td>Canada</td>
  </tr>
  <tr>
    <td>Magazzini Alimentari Riuniti</td>
    <td>Giovanni Rovelli</td>
    <td>Italy</td>
  </tr>
  <tr>
    <td>North/South</td>
    <td>Simon Crowther</td>
    <td>UK</td>
  </tr>
  <tr>
    <td>Paris spécialités</td>
    <td>Marie Bertrand</td>
    <td>France</td>
  </tr>
</table>

</body>
</html>
```

A Fancy Table

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Berglunds snabbköp	Christina Berglund	Sweden
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Königlich Essen	Philip Cramer	Germany
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada
Magazzini Alimentari Riuniti	Giovanni Rovelli	Italy
North/South	Simon Crowther	UK
Paris spécialités	Marie Bertrand	France

https://www.w3schools.com/cssref/default.asp

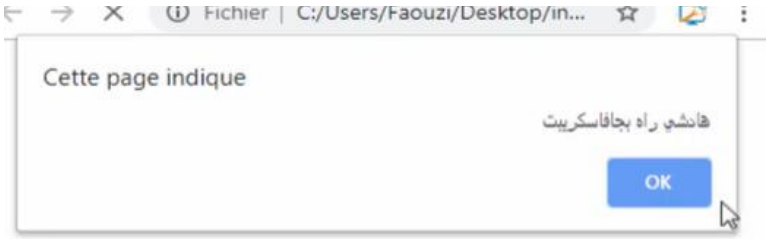
JS

JavaScript

Vaut mieux écrire le code dans un éditeur de texte avec l'extension .js

<script> ---------- </script>

```
<title>Où écrire le code javascript
<script>
    alert("هادشي راه بجافاسكريبت");
    //c'est un commentaire en javas
</script>
```



Pour savoir si js est activé ou pas sur notre navigateur

<noscript> Le message qu'on veut afficher</noscript>

L'opérateur ternaire

```
"Le prix est : " + (estMembre ? "15 €" : "30 €")
```

```
var elvisLives = Math.PI > 4 ? "Yep" : "Nope";
```

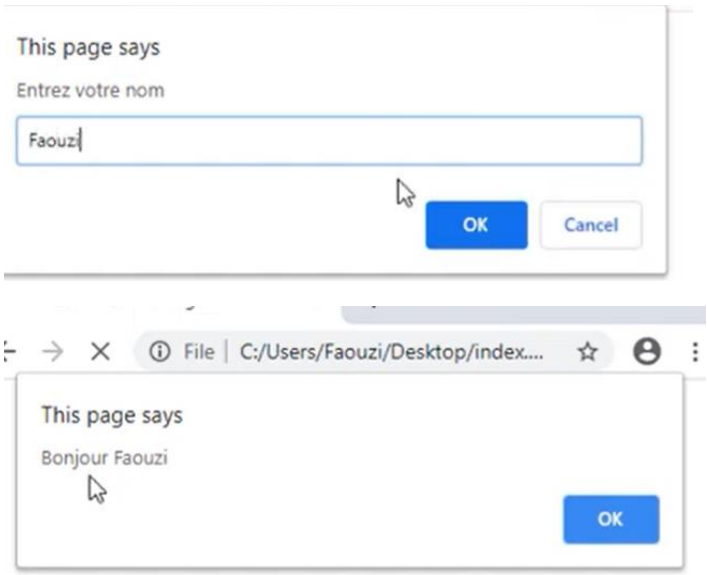
Pour afficher sur la console

Console.log(ce qu'on veut afficher) ;

Prompt

- affiche une zone de texte

```
<head>
  <meta charset="UTF-8">
  <title>Les boites de dialogue</title>
    <script>
        //alert("Bonjour cher utilisateur");
        var nom = prompt("Entrez votre nom");
        alert("Bonjour "+nom);
    </script>
```



Confirm('texte') : retourne true or false

NaN : pour les erreurs

tab[tab.length]=" " : pour ajouter des elts à la fin d'un tab

ou tab.push(" ") ;

tab.unshift(" ") : ajout au début

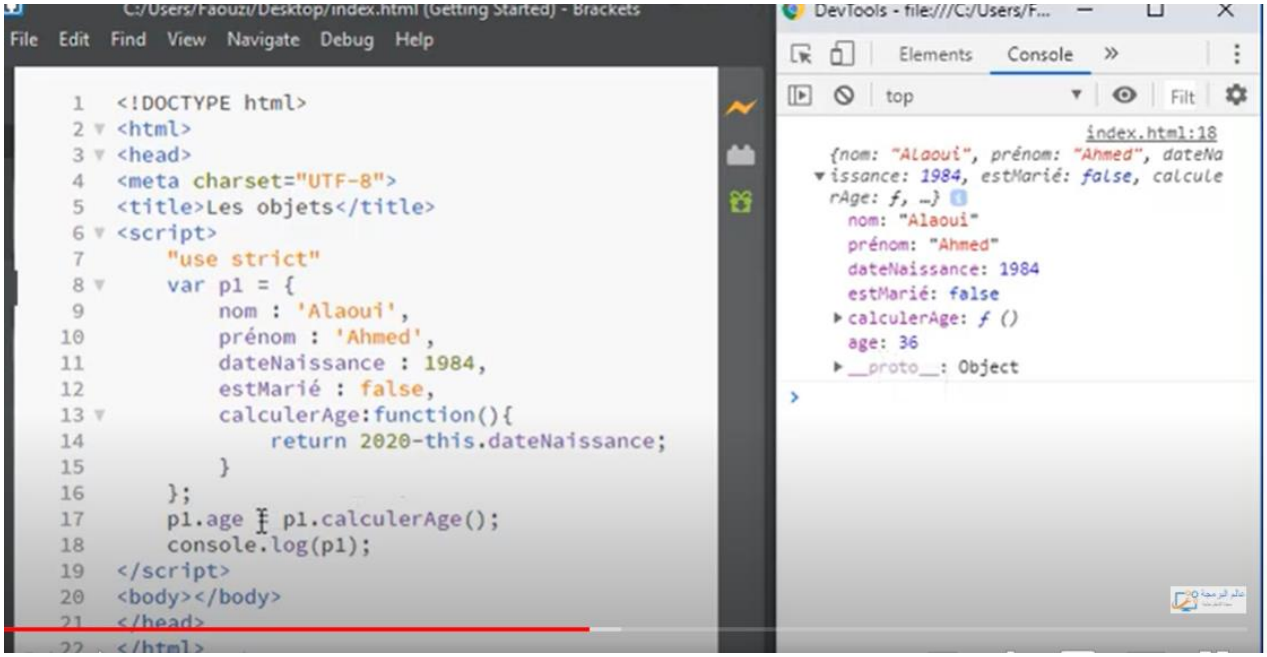
slice : créer un tab d'un autre tab sans modification

splice : créer un tab d'un autre tab avec modification

Les objets

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Les objets: Introduction</title>
<script>
    "use strict"
    var p1 = {
        nom : 'Alaoui',
        prénom : 'Ahmed',
        dateNaissance : 1984,
        estMarié : false,
    };
    console.log(p1.nom);
    console.log(p1['prénom']);
    var date = 'dateNaissance';
    console.log(p1[date]);
</script>
<body></body>
</head>
</html>
```

var p2 = new Object() ;



https://www.w3schools.com/jsref/default.asp

```html
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</head>
<body>

<h2>Demo JavaScript in Head</h2>

<p id="demo">A Paragraph.</p>

<button type="button" onclick="myFunction()">Try it</button>

</body>
</html>
```

Demo JavaScript in Head

A Paragraph.

Try it

After clicking on « Try it »

Demo JavaScript in Head

Paragraph changed.

Try it

Keyword	Description
var	Declares a variable
let	Declares a block variable
const	Declares a block constant
if	Marks a block of statements to be executed on a condition
switch	Marks a block of statements to be executed in different cases
for	Marks a block of statements to be executed in a loop
function	Declares a function
return	Exits a function
try	Implements error handling to a block of statements

```
let x, y, z;
```

Fixed values are called **Literals**.

Variable values are called **Variables**.

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type

Operator	Description
typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type

Instanceof “Object”

Exple:

Instanceof Array

typeof 3

typeof “salma”

Objects

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

The **name:values** pairs in JavaScript objects are called **properties**

Accessing Object Properties :

objectName.propertyName

or

objectName["propertyName"]

exple :

```
person.lastName;
```

```
person["lastName"];
```

```
fullName : function() {
    return this.firstName + " " + this.lastName;
  }
```

```
name = person.fullName();
```

Events

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

https://www.w3schools.com/jsref/dom_obj_event.asp

Strings

https://www.w3schools.com/jsref/jsref_obj_string.asp

Both `indexOf()`(first occ of a text), **and** `lastIndexOf()`(last occ of a text) **return -1 if the text is not found**

```
let str = "Please locate where 'locate' occurs!";
str.lastIndexOf("locate");
```

output : 21

Both methods accept a second parameter as the starting position for the search: `str.indexOf("locate", 15);`

The `startsWith()` **method returns** `true` **if a string begins with a specified value, otherwise** `false`

`endsWith().`

Numbers

`isNaN()` **to find out if a value is a not a number**

https://www.w3schools.com/jsref/jsref_obj_number.asp

Arrays

The `sort()` **method sorts an array alphabetically**

https://www.w3schools.com/jsref/jsref_obj_array.asp

Date

```
const d = new Date();
```

JavaScript counts months from `0` **to** `11`**:**

https://www.w3schools.com/jsref/jsref_obj_date.asp

Type	Example
ISO Date	"2015-03-25" (The International Standard)
Short Date	"03/25/2015"
Long Date	"Mar 25 2015" or "25 Mar 2015"

Math

https://www.w3schools.com/jsref/jsref_obj_math.asp

Loops

- **for - loops through a block of code a number of times**
- **for/in - loops through the properties of an object**
- **for/of - loops through the values of an iterable object**
- **while - loops through a block of code while a specified condition is true**
- **do/while - also loops through a block of code while a specified condition is true**

```
for (let x in person) {
  text += person[x];
}
```

```
let language = "JavaScript";

let text = "";

for (let x of language) {

  text += x + "<br>";

}
```

Output:

J
a
v
a
S
c
r
i
p
t

JQuery

Return the element with id="id01":

```
myElement = $("#id01");
```

Return all <p> elements:

```
myElements = $("p");
```

Return all elements with class="intro".

```
myElements = $(".intro");
```

php

(Hypertext preprocessor)

PHP is a widely-used, open-source scripting language

PHP scripts are executed on the server

What Do I Need?

To start using PHP, you can:

Find a web host with PHP and MySQL support

Install a web server on your own PC, and then install PHP and MySQL

Syntax

```
<?php
// PHP code goes here
?>
```

Note: PHP statements end with a semicolon (`;`).

With PHP, there are two basic ways to get output: `echo` and `print`.

echo "message";

echo nom_variable;

The echo statement can be used with or without parentheses: echo or echo()

PHP stores all global variables in an array called $GLOBALS[*index*].
The *index* holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.

```
<?php
$x = 5;
$y = 10;

function myTest() {
  $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}

myTest();
echo $y; // outputs 15
?>
```

Variables

The PHP **var_dump()** function returns the data type and value

```
<?php
$x = 5985;
var_dump($x);
?>
```

Output :

```
int(5985)
```

```
$x = 10.365;
```

Output :

```
float(10.365)
```

```
$cars = array("Volvo","BMW","Toyota");
```

Output :

```
array(3) {
  [0]=>
  string(5) "Volvo"
  [1]=>
  string(3) "BMW"
  [2]=>
  string(6) "Toyota"
}
```

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

<?php
class Car {
  public $color;
  public $model;
  public function __construct($color, $model) {
    $this->color = $color;
    $this->model = $model;
  }
  public function message() {
    return "My car is a " . $this->color . " " . $this->model . "!";
  }
}

$myCar = new Car("black", "Volvo");
echo $myCar -> message();
echo "<br>";
$myCar = new Car("red", "Toyota");
echo $myCar -> message();
?>

</body>
</html>
```

My car is a black Volvo!
My car is a red Toyota!

Strings

The PHP `strlen()` function returns the length of a string

The PHP `str_word_count()` function counts the number of words in a string

The PHP `strrev()` function reverses a string.

The PHP `strpos()` function searches for a specific text within a string. If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.

```
echo strpos("Hello world!", "world"); // outputs 6
```

The PHP `str_replace()` function replaces some characters with some other characters in a string

```
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!
```

Integer

PHP has the following functions to check if the type of a variable is integer:

is_int()

is_integer() - alias of is_int()

is_long() - alias of is_int()

outputs:

bool(true)
bool(false)

PHP has the following functions to check if the type of a variable is float:

is_float()

is_double() - alias of is_float()

is_nan()

The PHP is_numeric() function can be used to find whether a variable is numeric. The function returns true if the variable is a number or a numeric string, false otherwise.

(function will return FALSE for numeric strings in hexadecimal form (e.g. 0xf4c3b00c))

Constants

A valid constant name starts with a letter or underscore (no $ sign before the constant name).

Note: Unlike variables, constants are automatically global across the entire script.

Syntax

`define(name, value, case-insensitive)`

- *case-insensitive*: Specifies whether the constant name should be case-insensitive. Default is false

```
define("GREETING", "Welcome to W3Schools.com!", true);
```

Operators

<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.

Returns -1,0,1

returns -1 if $x is less than $y

.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2

A regular expression is a sequence of characters that forms a search pattern. When you search for data in a text, you can use this search pattern to describe what you are searching for.

PHP Forms

1-Handling

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

Run Example »

When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method.

To display the submitted data you could simply echo all the variables. The "welcome.php" looks like this:

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>
```

The output could be something like this:

```
Welcome John
Your email address is john.doe@example.com
```

The same result could also be achieved using the HTTP GET method:

```
<form action="welcome_get.php" method="get">
```

and "welcome_get.php" looks like this:

```
<html>
<body>

Welcome <?php echo $_GET["name"]; ?><br>
Your email address is: <?php echo $_GET["email"]; ?>

</body>
</html>
```

The code above is quite simple. However, the most important thing is missing. You need to validate form data to protect your script from malicious code.

The code above is quite simple. However, the most important thing is missing. You need to validate form data to protect your script from malicious code.

GET vs. POST

Both GET and POST create an array (e.g. array(key1 => value1, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.

Both GET and POST are treated as $_GET and $_POST. These are superglobals, which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

$_GET is an array of variables passed to the current script via the URL parameters.

$_POST is an array of variables passed to the current script via the HTTP POST method.

When to use GET?

Information sent from a form with the GET method is **visible to everyone** (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

GET may be used for sending non-sensitive data.

Note: GET should NEVER be used for sending passwords or other sensitive information!

When to use POST?

Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send.

Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.

However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

Developers prefer POST for sending form data.

2-Validation

The first thing we will do is to pass all variables through PHP's htmlspecialchars() function.

When we use the htmlspecialchars() function; then if a user tries to submit the following in a text field:

<script>location.href('http://www.hacked.com')</script>

- this would not be executed, because it would be saved as HTML escaped code, like this:

<script>location.href('http://www.hacked.com')</script>

The code is now safe to be displayed on a page or inside an e-mail.

We will also do two more things when the user submits the form:

1. Strip unnecessary characters (extra space, tab, newline) from the user input data (with the PHP trim() function)
2. Remove backslashes (\) from the user input data (with the PHP stripslashes() function)

The next step is to create a function that will do all the checking for us (which is much more convenient than writing the same code over and over again).

We will name the function test_input().

Now, we can check each $_POST variable with the test_input() function, and the script looks like this:

```php
<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>
```

at the start of the script, we check whether the form has been submitted using $_SERVER["REQUEST_METHOD"]. If the REQUEST_METHOD is POST, then the form has been submitted - and it should be validated. If it has not been submitted, skip the validation and display a blank form.

3-Required

From the validation rules table on the previous page, we see that the "Name", "E-mail", and "Gender" fields are required. These fields cannot be empty and must be filled out in the HTML form.

Field	Validation Rules
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

In the following code we have added some new variables: $nameErr, $emailErr, $genderErr, and $websiteErr. These error variables will hold error messages for the required fields. We have also added an `if else` statement for each $_POST variable. This checks if the $_POST variable is empty (with the PHP `empty()` function). If it is empty, an error message is stored in the different error variables, and if it is not empty, it sends the user input data through the `test_input()` function:

```php
<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}
?>
```

Example

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">

Name: <input type="text" name="name">
<span class="error">* <?php echo $nameErr;?></span>
<br><br>
E-mail:
<input type="text" name="email">
<span class="error">* <?php echo $emailErr;?></span>
<br><br>
Website:
<input type="text" name="website">
<span class="error"><?php echo $websiteErr;?></span>
<br><br>
Comment: <textarea name="comment" rows="5" cols="40"></textarea>
<br><br>
Gender:
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male
<input type="radio" name="gender" value="other">Other
<span class="error">* <?php echo $genderErr;?></span>
<br><br>
<input type="submit" name="submit" value="Submit">

</form>
```

PHP Form Validation Example

* required field

Name: * Name is required

E-mail: * Email is required

Website:

Comment:

Gender: ○ Female ○ Male ○ Other * Gender is required

Submit

Your Input:

PHP - Validate Name

The code below shows a simple way to check if the name field only contains letters, dashes, apostrophes and whitespaces. If the value of the name field is not valid, then store an error message:

```
$name = test_input($_POST["name"]);
if (!preg_match("/^[a-zA-Z-' ]*$/",$name)) {
  $nameErr = "Only letters and white space allowed";
}
```

The preg_match() function searches a string for pattern, returning true if the pattern exists, and false otherwise.

PHP - Validate E-mail

The easiest and safest way to check whether an email address is well-formed is to use PHP's filter_var() function.

In the code below, if the e-mail address is not well-formed, then store an error message:

```
$email = test_input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
  $emailErr = "Invalid email format";
}
```

PHP - Validate URL

The code below shows a way to check if a URL address syntax is valid (this regular expression also allows dashes in the URL). If the URL address syntax is not valid, then store an error message:

```
$website = test_input($_POST["website"]);
if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
  $websiteErr = "Invalid URL";
}
```

```
<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
    // check if name only contains letters and whitespace
    if (!preg_match("/^[a-zA-Z-' ]*$/",$name)) {
      $nameErr = "Only letters and white space allowed";
    }
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
    // check if e-mail address is well-formed
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $emailErr = "Invalid email format";
    }
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
    // check if URL address syntax is valid
    if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-
```

```
9+&@#\/%=~_|]/i",$website)) {
      $websiteErr = "Invalid URL";
    }
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  E-mail: <input type="text" name="email">
  <span class="error">* <?php echo $emailErr;?></span>
  <br><br>
  Website: <input type="text" name="website">
  <span class="error">><?php echo $websiteErr;?></span>
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" value="female">Female
  <input type="radio" name="gender" value="male">Male
  <input type="radio" name="gender" value="other">Other
  <span class="error">* <?php echo $genderErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```

4-FORM Completed

PHP - Keep The Values in The Form

To show the values in the input fields after the user hits the submit button, we add a little PHP script inside the value attribute of the following input fields: name, email, and website. In the comment textarea field, we put the script between the <textarea> and </textarea> tags. The little script outputs the value of the $name, $email, $website, and $comment variables.

Then, we also need to show which radio button that was checked. For this, we must manipulate the checked attribute (not the value attribute for radio buttons):

```
Name: <input type="text" name="name" value="<?php echo $name;?>">

E-mail: <input type="text" name="email" value="<?php echo $email;?>">

Website: <input type="text" name="website" value="<?php echo $website;?
>">

Comment: <textarea name="comment" rows="5" cols="40"><?php echo $commen
t;?></textarea>

Gender:
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="female") echo "checked";?>
value="female">Female
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="male") echo "checked";?>
value="male">Male
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="other") echo "checked";?>
value="other">Other
```

https://www.w3schools.com/php/php_ref_overview.asp