Javascript

DHTML

But du DHTML

```html
<!doctype html>
<html lang="fr">
<head>
   <meta charset="utf-8">
   <title>DHTML</title>
   <link rel="stylesheet" href="styles.css">
</head>
<body>
<h1 id = "para1">Hello world !</h1>
<input class="btn" type = "Submit" onclick = "Click()"/>
<script src="script.js"></script>
</body>
</html>
```

```javascript
function Click() {
   document.getElementById("para1").style.color = "#009900";
   window.alert("La couleur est passée au vert !");
}
```

DHTML ou Dynamique HyperText Markup Language permet de rendre dynamique une page HTML sans avoir à la rechargée et ce grâce à un écouteur d'événements.

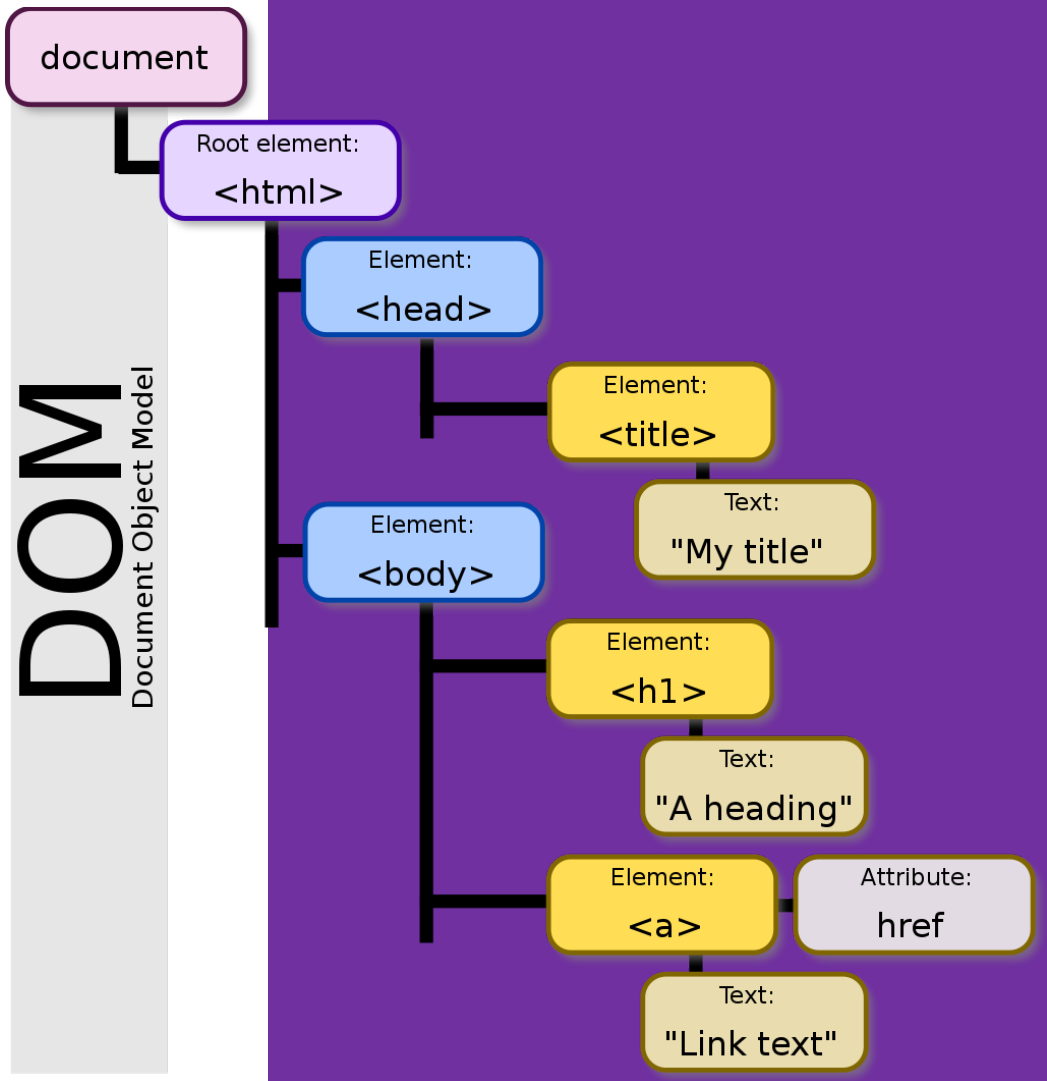
On déclare l'attribut « onclick » sur l'élément que l'on écoute. On y mets la fonction que l'on vas exécuter en js quand l'action est exécutée par l'utilisateur.

Dans le fichier « script.js », on déclare la fonction avec « function Click(){ }. Dans les accolades, on met le code que la fonction doit exécuter. Ici, dans cet exemple, le script vas récuperer l'élément qui a l'id « para1 » pour lui change la couleur de la police d'écriture. Puis, pour finir, on affiche une fenêtre d'alerte avec le texte suivant: « La couleur est passée au vert !».

Présentation du DOM

Le DOM ou Document Object Model a été mis en place par le W3C pour déterminer et représenter la structure d'un document HTML sous forme d'objet.

Cet objet est lisible en JavaScript pour être plus facilement modifiable. Dans cet structure, chaque élément est une représentation en Objet d'un élément en HTML.



Attributs DHTML

Voici un exemple d'attributs qui servent d'écouteur lors d'un événement:

onabort: Se déclenche lorsque l'utilisateur stoppe le chargement de la page.

onblur: Se déclenche lorsqu'un élément de la page perd le focus (lorsque l'utilisateur clique ailleurs).

onchange: Se déclenche à chaque fois que la valeur d'un champ de donnée voit sa valeur modifiée.

onclick: Se déclenche lorsque l'utilisateur clique sur l'élément en question.

onkeypress: Se déclenche lorsque l'utilisateur maintient une touche du clavier enfoncée.

onload: Se déclenche lorsque l'utilisateur charge la page.

onmousedown: Se déclenche lorsque l'utilisateur appuie sur un bouton de sa souris au-dessus d'un élément de la page

Manipulation des noeuds du DOM

Un nœud dans le DOM c'est un élément unique qui peut être le document lui-même, du texte, un titre, etc...

Il est possible de sélectionner le document en lui-même avec « document » mais aussi de sélectionner un élément par son « id », son nom, sa classe ou la balise HTML grâce aux fonctions montrées dans cet exemple.

```
document.getElementById('btn')
document.getElementsByName('name')
document.getElementsByClassName('class')
document.getElementsByTagName('div')
```

Gestion des événements

```
function validation(e){

}
```

Dans cette fonction « validation », le paramètre « e » fait référence à l'événement qui vas se propager dans cette fonction. Par exemple si il s'agit de l'envoi d'un formulaire, il est possible de l'arrêter.

```
img.addEventListener('mouseenter',
function (e) {
    img.setAttribute("src",
"./assets/img/2.jpg")
 })
```

Pour les événements, il est aussi possible de mettre un écouteur d'événements avec la fonction « addEventListener » sur l'élément que l'on veut écouter. Dans cet exemple, l'élément « img » réagira sur une entrée de la souris.

Javascript

Les formulaires

```html
<!doctype html>
<html lang="fr">
<head>
  <meta charset="utf-8">
  <title>Validation de formulaires</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <form method="post" action="execution.php">
    <label for="">Veuillez entrer votre prénom:</label>
    <input type="text" name="prenom" id="prenom"
maxlength="20" required>
    <span id="erreur_prenom"></span><br>

    <label for="">Veuillez entrer votre mail:</label>
    <input type="email" name="mail" id="mail"
required> <br>

    <label for="">Veuillez entrer votre numéro de
téléphone:</label>
    <input type="tel" name="phone" id="phone"
required> <br>

    <input type="submit" id="envoi" value="Envoyer">
  </form>
  <script src="script.js"></script>
</body>
</html>
```

Rappel des formulaires HTML

La validation en HTML se fait avec l'attribut « required » pour spécifier que le champ doit être obligatoirement rempli par l'utilisateur pour valider le formulaire.

D'autres attributs comme « minlength » ou « maxlength » permettent de définir un minimum et un maximum de caractères à ne pas dépasser lorsque l'on remplit le champ.

Malheureusement, la validation en HTML n'est pas assez efficace pour contrôler les formulaires. En effet, les utilisateurs mal attentionnés peuvent enlever l'attribut et forcer la validation du formulaire. C'est pour cela qu'il est préférable de faire des vérifications en JavaScript puis en PHP.

Les événements associés

Voici quelques événements qui sont généralement associés a un formulaire:

onBlur	*Un élément du formulaire qui n'est plus actif.*
onChange	*Lors d'un changement sur un select, un texte ou un textarea.*
onClick	*Au clic.*
onFocus	*Quand l'élément est sélectionné au clavier.*
onReset	*Quand le formulaire est remis à zero.*
onSelect	*Quand le texte est sélectionné.*
onSubmit	*Quand le formulaire est soumis.*

La collection des objets d’un formulaire

```
console.log(document.forms)
```

Le formulaire peut se voir comme un objet grâce à « document.forms » qui récupère tous les formulaires d’une page.

Ensuite, il est possible de manipuler plus facilement un formulaire pour modifier un champ dynamiquement ou pour faire de la vérification.

Vérification des champs

La validation en JavaScript permet de faire une vérification des champs plus poussée tout en évitant d'appeler un script côté serveur.

Dans cet exemple, l'on vérifie que le champ ne soit pas vide ou qu'il ne correspond pas à une expression régulière (regex).

Il est possible pour l'utilisateur de désactiver JavaScript sur sa machine (ce cas de figure reste très marginal), il faut donc faire une dernière vérification côté serveur pour écarter tout risques.

```
var prenom = document.getElementById('prenom');
var erreur_prenom =
document.getElementById('erreur_prenom');
var prenom_regex = /^[a-zA-ZéèîïÉÈÎÏ][a-zéèîïçà]+([-
'\s][a-zA-ZéèîïÉÈÎÏ][a-zéèîïçà]+)?/;


   if(prenom.validity.valueMissing){
      e.preventDefault();
      erreur_prenom.textContent = 'Il manque votre
prénom !';
      erreur_prenom.style.color = 'red';
   }else if(prenom_regex.test(prenom.value) ===
false){
      e.preventDefault();
      erreur_prenom.textContent = 'Le format de
votre prénom est incorrect';
      erreur_prenom.style.color = 'orange';
   }
```

Vérification globale d’un formulaire

```
var validate = document.getElementById('envoi');
var prenom = document.getElementById('prenom');
var erreur_prenom =
document.getElementById('erreur_prenom');
var prenom_regex = /^[a-zA-ZéèîïÉÈÎÏ][a-zéèîïçà]+([-
'\s][a-zA-ZéèîïÉÈÎÏ][a-zéèîïçà]+)?/;

validate.addEventListener('click', validation);

function validation(e){
   if(prenom.validity.valueMissing){
      e.preventDefault();
      erreur_prenom.textContent = 'Il manque votre
prénom !';
      erreur_prenom.style.color = 'red';
   }else if(prenom_regex.test(prenom.value) ===
false){
      e.preventDefault();
      erreur_prenom.textContent = 'Le format de
votre prénom est incorrect';
      erreur_prenom.style.color = 'orange';
   }
}
```

Une fois que la validation des champs est mise en place, il est possible de vérifier tous les champs du formulaire et ainsi de décider si le formulaire est bien rempli dans sa globalité.

La fonction « preventDefault » permet de bloquer l’événement qui déclenche la validation du formulaire.

Javascript

Travailler avec les images

Propriétés de l’objet image

```
function changerImage(img, src, maxWidth, maxHeight)
{
   // on crée l'objet
   var image = new Image();

   // événements : cas d'erreur
   image.onerror = function()
   {
      alert("Erreur lors du chargement de l'image");
   };
   image.onabort = function()
   {
      alert("Chargement interrompu");
   };

}
```

En JavaScript, il est aussi possible de manipuler des images avec l’objet dédié: « image ».

Dans le début de cette fonction qui comprends plusieurs paramètres (l’image, la source, la taille maxi et la hauteur maximum), on initialise l’objet « image » pour manipuler l’image passée dans cette fonction.

Ensuite, l’on vérifie que l’image s’est bien chargée. Si ce n’est pas le cas, l’on affiche à l’utilisateur un message d’avertissement.

La collection des objets images

Dans la suite de l'exemple, le cœur de la fonction se mets en route une fois que l'image est chargée. Elle permet de redimensionner l'image si elle est trop grande par rapport aux dimensions spécifiées dans les paramètres de la fonction.

La fonction « Math.max » permet de renvoyer le plus grand nombre d'une série. Et « Math.round » permet de renvoyer l'entier le plus proche. Une fois que la réduction a été faite, l'on affiche à l'utilisateur l'image qui a été réduite si besoin.

La dernière ligne représente l'appel fait à la fonction « changerImage » en sélectionnant l'image par son id tout en précisant le type, la largeur maximale et la hauteur maximale.

```
// événement : une fois le chargement terminé
image.onload = function()
{
   // si l'image est désignée par son id
   if(typeof img == "string")
      img = document.getElementById(img);

   // si l'image doit être redimensionnée
   var reduction = 1;
   if(maxWidth && maxHeight)
      if(image.width > maxWidth || image.height > maxHeight)
         reduction = Math.max(image.width/maxWidth,
image.height/maxHeight);

   // on affiche l'image
   img.src = image.src;
   img.width = Math.round(image.width / reduction);
   img.height = Math.round(image.height / reduction);

};

// on modifie l'adresse de l'objet "image", ce qui lance le chargement
image.src = src;
```

```
changerImage(document.getElementById("ImageTest"),
"images/baltic-sea-1920.png", 400, 400);
```

Précharger des images

```
function load() {
    this.length=load.arguments.length;
    for (var i=0;i<this.length;i++) {
        this[i+1]=new Image();
        this[i+1].src=load.arguments[i];
    }
}
function preload() {
    // Cette fonction charge dans le cache toutes
 les images passées en paramètre
    load("image1.gif","image2.gif","image3.gif")
}
```

```
<body onLoad="preload()">
```

Pour précharger des images, on crée une fonction « load » qui vas mettre en place les images passées en argument dans la fonction « preload » avec l'objet image de JavaScript.

Ensuite la fonction est appelée dans l'attribut « onLoad » de la balise « body » de la page HTML.

Réaliser un rollover

Pour créer un « rollover » ou changer une image au survol de celle-ci, il faut écouter deux événements.

Un quand la souris passe dans l'élément et un quand la souris sort de l'élément. Après il ne reste plus qu'a changer le src de l'imge en question pour pouvoir changer l'image.

```
<img src="./assets/img/1.jpg" id="img">

<script>
   var img = document.getElementById('img')
   img.addEventListener('mouseenter', function (e) {
      img.setAttribute("src", "./assets/img/2.jpg")
   })
   img.addEventListener('mouseleave', function (e){
      img.setAttribute("src", "./assets/img/1.jpg")
   })
</script>
```

Création d’un diaporama(HTML)

```
<div class="frame-slide">
   <img class="slide" src="img/1.jpg" alt="">
   <img class="slide" src="img/2.jpg" alt="">
   <img class="slide" src="img/3.jpg" alt="">
   <button class="prev" aria-label="prev"
onclick="buttons(-1)">❮</button>
   <button class="next" aria-label="next"
onclick="buttons(1)">❯</button>
</div>
<div class=frame-indicator>
   <button class="indicator"
onclick="activateIndicator(1)">1</button>
   <button class="indicator"
onclick="activateIndicator(2)">2</button>
   <button class="indicator"
onclick="activateIndicator(3)">3</button>
</div>
```

Avant de toucher au JavaScript pour créer le diaporama, il faut mettre en place la structure HTML.

Avec les éléments qui nous permettront de créer les différents événements au clic de l’utilisateur.

Et la structure qui accueille toutes les images du diaporama.

Les événements du diaporama

Ensuite, il faut travailler cette structure HTML avec du JavaScript.

La fonction « display(slide) » initialise le diaporama sur la première slide lors du chargement de la page HTML.

Ici, on mets en place les fonctions qui seront appelées au clic des boutons sur le côté ou sur les indicateurs des différentes slides.

```
var slide = 1;
display(slide);

function buttons(n) {
    display(slide += n);
}

function activateIndicator(n) {
    display(slide = n);
}
```

Diaporama, function d'affichage

```
function display(n) {
    var i;
    var slideImg = document.getElementsByClassName("slide");
    var indicators =
document.getElementsByClassName("indicator");
    if (n > slideImg.length) {slide = 1}
    if (n < 1) {slide = slideImg.length}
    for (i = 0; i < slideImg.length; i++) {

        slideImg[i].style.opacity = "0";
    }
    for (i = 0; i < indicators.length; i++) {
        indicators[i].className = indicators[i].className.replace("
numbers", "");
    }
    slideImg[slide-1].style.opacity = "1";
    indicators[slide-1].className += " numbers";
}
```

Comme vous l'avez surement remarquer juste avant, toutes les fonctions appellent la fonction « display » pour générer l'affichage de l'image du diaporama.

A chaque passage sur une image, le JavaScript change l'opacité de l'image pour la faire apparaître et change la couleur de l'indicateur correspondant.

Javascript

Les cookies

Ecrire un cookie

```
document.cookie =
"name=Sylvain;";
document.cookie = "work=helper;
expires=Thu, 05 Oct 2021
12:00:00 UTC; path=/;";

// retourne tous les cookies dans
une chaîne de caracteres
console.log(document.cookie);

document.cookie = "work=lead;
expires=Thu, 06 Oct 2021
12:00:00 UTC; path=/;";
```

Les cookies : ce sont des fichiers textes de petite taille qui sont stockés sur la machine de l'utilisateur. Ils permettent d'enregistrer des informations spécifiques sur le visiteur. Mais ils permettent aussi de maintenir une session pendant un laps de temps donné et ainsi permettre à l'utilisateur de revenir sur un site sans avoir besoin de se reconnecter. Même si ils sont souvent décriés car utilisés dans une grande partie des cas pour suivre les utilisateurs. Ils restent essentiel pour des sites d'e-commerce ou des sites proposants un espace de connexion.

Lire un cookie

Il peut aussi être nécessaire de le faire lors d'une interaction avec l'utilisateur. On utilise le « document.cookie » pour afficher tous les cookies utilisés à l'instant. Pour créer un nouveau cookie, il suffit de mettre des données dans « document.cookie » comme pour les variables.

Le nom, l'expiration ou encore la page qui est liée au cookie sont paramétrables. Pour le supprimer, il faut changer la date d'expiration et mettre une date antérieure à celle du jour.

```
console.log(document.cookie);

document.cookie = "work=lead;
expires=Thu, 01 Jan 2022
00:00:00 UTC; path=/;";

console.log(document.cookie);
```