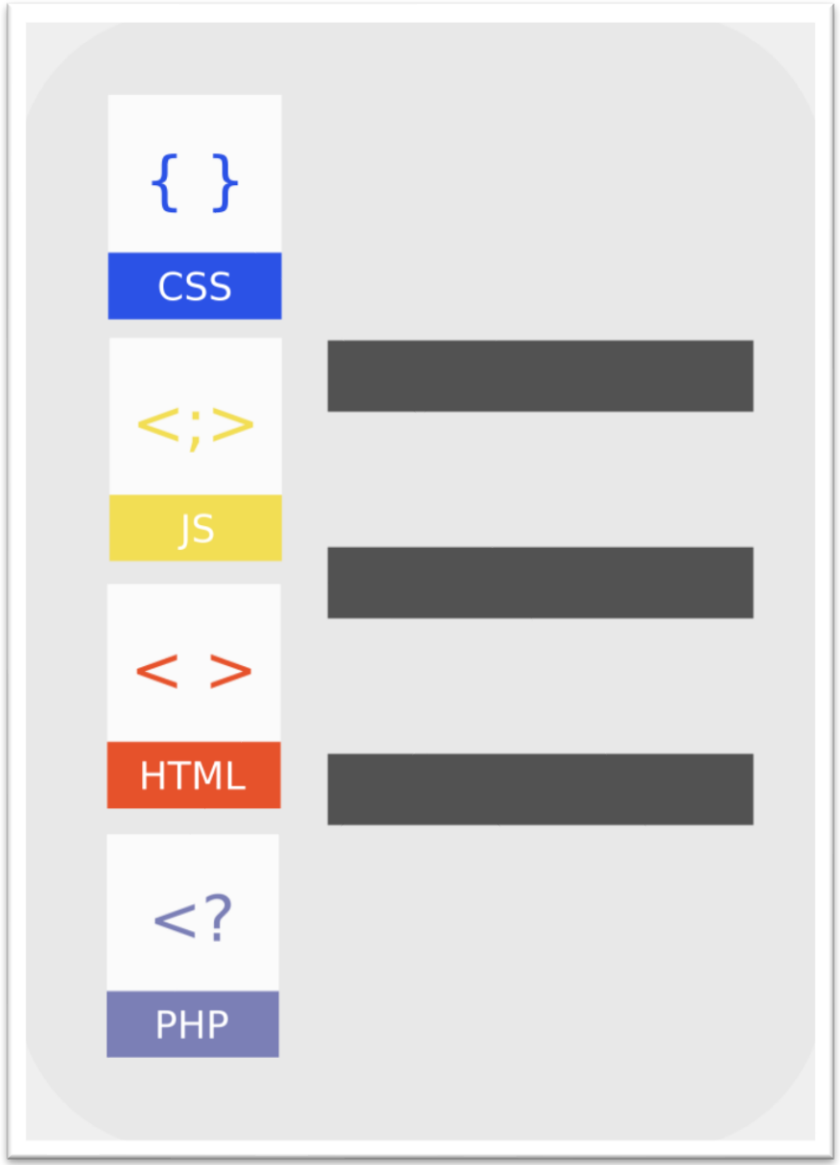
RUSH 1 - KICKOFF

Bases HTML, CSS, Javascript, serveurs, sécurité...

CREATION DE PAGE WEB

Lors de la création d'une page web, il est possible de mélanger du html, du css, du javascript et du php dans une même page.

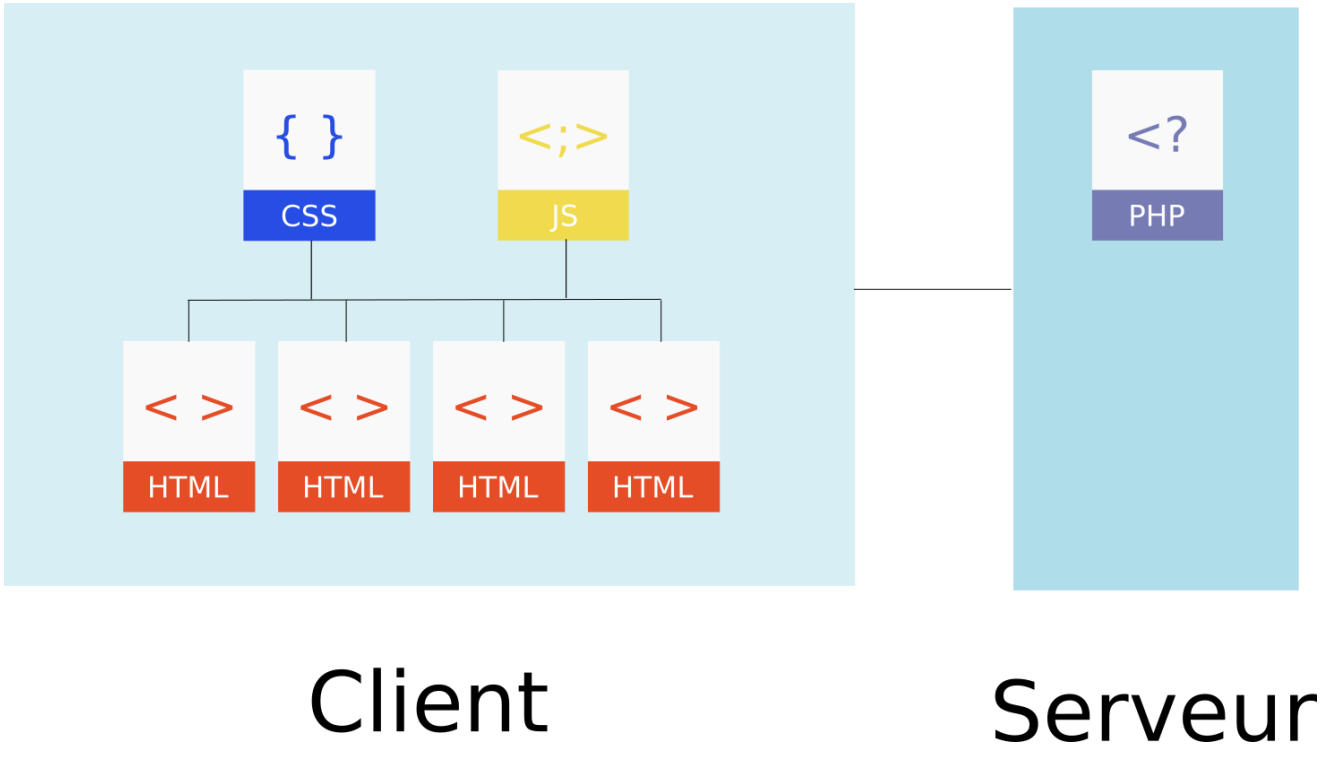


EXEMPLE DE CODE

```
<html>
<head>
    <style>
        #demo{
                color:green;
        }
    </style>
</head>
<body>

    <h2>What Can JavaScript Do?</h2>

    <p id="demo">JavaScript can change HTML content.</p>

    <button type="button" onclick='document.getElementById("demo").innerHTML = "Hello
    JavaScript!"'>Click Me!</button>

</body>
</html>
```

CREATION DE PAGE WEB

Pourquoi séparer son code dans des fichiers dédiés ?



CREATION DE PAGE WEB

Avantages de la séparation des fichiers :

- Retrouver facilement les informations
- Améliorer la cohérence d'un site
- Rapidité d'affichage
- Code plus léger
- Optimisation moteur de recherche
- Etc...

CREATION DE PAGE WEB

Il existe des sites sur lesquels il est possible de vérifier la syntaxe de nos pages :

- JavaScript : http://jshint.com/
- CSS3 : https://jigsaw.w3.org/css-validator/
- HTML5 : https://validator.w3.org/
- PHP : https://github.com/squizlabs/PHP_CodeSniffer

Et pour s'autoformer : https://www.w3schools.com/ par exemple.

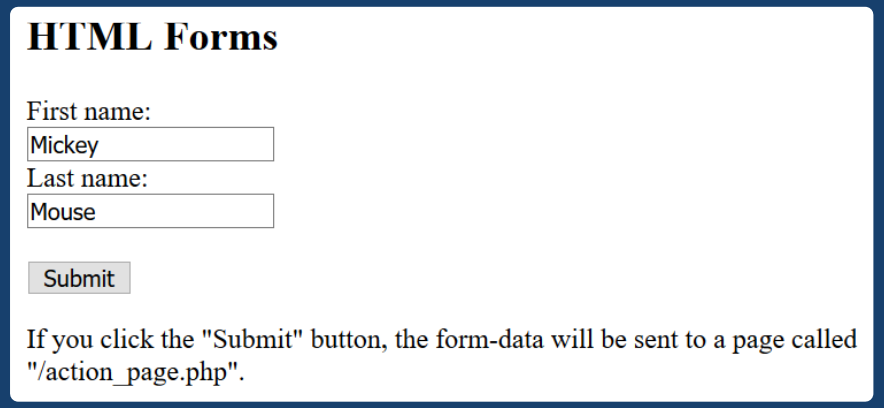
CREATION DE PAGE WEB

En quoi ces validateurs sont ils utiles ?

- Aide au debug
- Ajoute une vérification de la qualité du code
- Aide pour les futures maintenances
- Apprendre les bonnes pratiques

Vous devez utiliser ces validateurs en tous temps, vos pages pourront être validées par cette méthode en soutenance.

QU'EST QU'UN FORMULAIRE HTML ?




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

        <h2>HTML Forms</h2>

        <form action="/action_page.php">
            First name:<br>
            <input type="text" name="firstname"
            value="Mickey">
            <br>
            Last name:<br>
            <input type="text" name="lastname"
            value="Mouse">
            <br><br>
            <input type="submit" value="Submit">
        </form>

        <p>If you click the "Submit" button, the form-data
        will be sent to a page called
        "/action_page.php".</p>

    </body>
</html>
```

GET OU POST ?

Deux méthodes d'accès définies dans le protocole HTTP et reprises dans la spécification HTML.

- GET permet en général de récupérer des données, on utilise aussi GET pour afficher une URL particulière (dans le cas d'une recherche par exemple)
- POST permet d'en envoyer

GET

```
<form method="get" action="page.html">
```

Données encodées dans l'URL, possible de les récupérer en JAVASCRIPT.

```
https://www.mon-site.fr/page.html?couleur=bleu&forme=rectangle
```

Quand la requête n'implique pas de changement dans les données. Une simple lecture.

POST

```
<form method="post" action="page.php">
```

- Envoie d'un en-tête et un corps de message au serveur (les données entrées dans les champs de formulaire par l'utilisateur).
- Pas de limite de taille.
- Impossible de les récupérer en JAVASCRIPT, usage du PHP possible par exemple.

POST

```php
<?php
    $couleur = $_POST['couleur'];
    $forme = $_POST['forme'];
?>
... code HTML ...
```

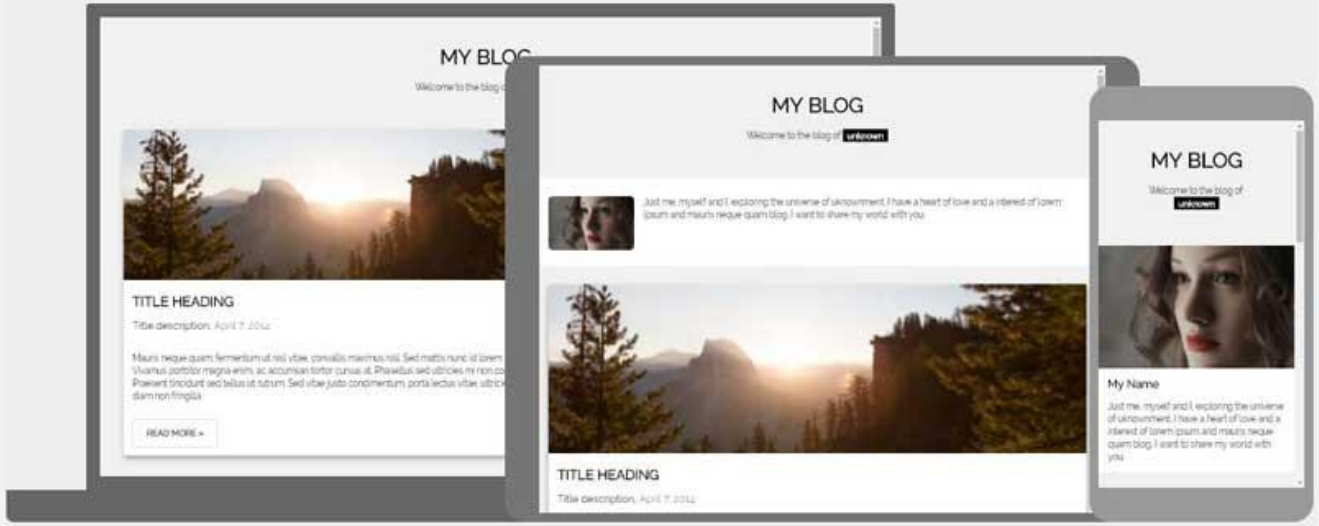
Possible de les assigner dans un script javascript :

```php
<script>
    var couleur = <?php echo $couleur;?>;
    var forme = <?php echo $forme;?>;
</script>
```

Ceci est un exemple de mauvaise pratique ! A ne pas reproduire chez vous.

CSS

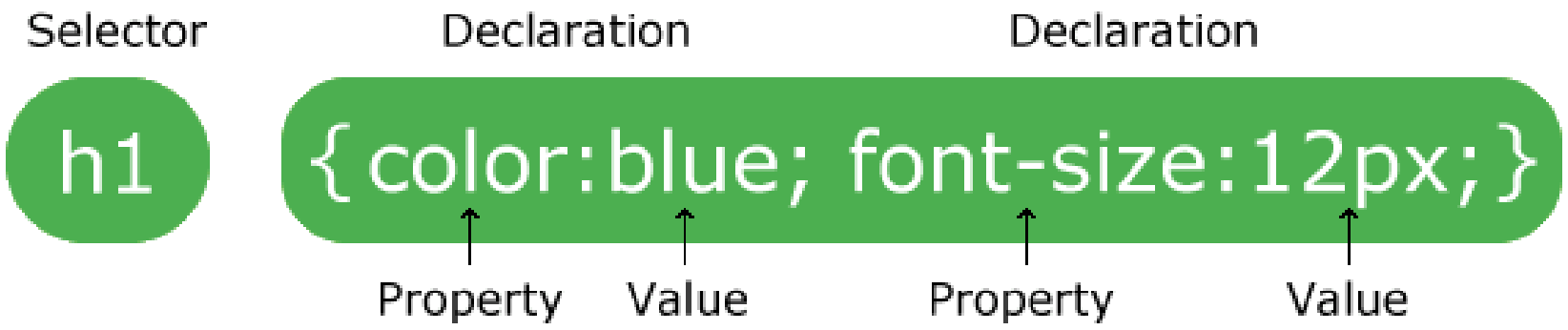
Décrire le style d'une page.



CSS

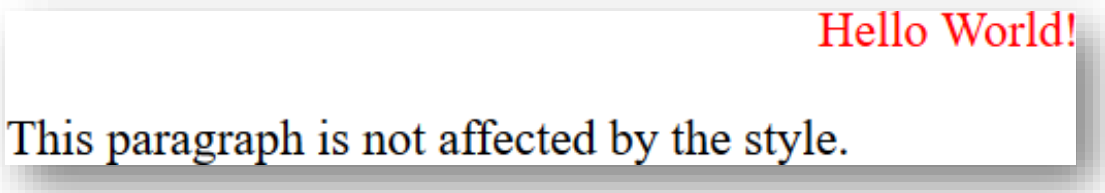
```css
body {
        background-color: lightblue;
}

h1 {
    color: white;
    text-align: center;
}

p {
    font-family: verdana;
    font-size: 20px;
}
```

CSS


Selector
Declaration
Declaration
h1
{color:blue; font-size:12px;}
Property
Value
Property
Value

CSS ID

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            #para1 {
                text-align: center;
                color: red;
            }
        </style>
    </head>
    <body>

        <p id="para1">Hello World!</p>
        <p>This paragraph is not affected by
        the style.</p>

    </body>
</html>
```

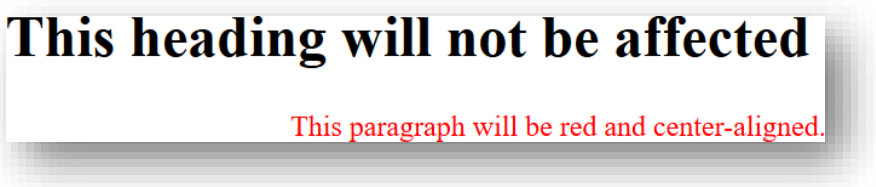


CSS CLASS

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            .center {
                text-align: center;
                color: red;
            }
        </style>
    </head>
    <body>

        <h1 class="center">Red and center-
        aligned heading</h1>
        <p class="center">Red and center-
        aligned paragraph.</p>

    </body>
</html>
```

CSS CLASS P

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            p.center {
                text-align: center;
                color: red;
            }
        </style>
    </head>
    <body>
        <h1 class="center">This heading will
        not be affected</h1>
        <p class="center">This paragraph will
        be red and center-aligned.</p>
    </body>
</html>
```



JAVASCRIPT

Manipuler des pages côté client.

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

    <h2>What Can JavaScript Do?</h2>

    <p id="demo">JavaScript can change HTML content.</p>

    <button type="button" onclick='document.getElementById("demo").innerHTML =
    "Hello JavaScript!"'>Click Me!</button>

    </body>
</html>
```

JAVASCRIPT

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

        <h2>JavaScript For Loop</h2>

        <p id="demo"></p>

        <script>
            var cars = ["BMW", "Volvo", "Saab", "Ford", "Fiat", "Audi"];
            var text = "";
            var i;
            for (i = 0; i < cars.length; i++) {
            text += cars[i] + "<br>";
            }
            document.getElementById("demo").innerHTML = text;
        </script>

    </body>
</html>
```

JAVASCRIPT

- Intérêt de séparer le code dans un fichier dédié (généricité).
- Pourvoir réutiliser le même code sur différentes pages quand nécessaire.

Et pourquoi pas une validation générique de formulaire…

```
function ValidateEmail(mail)
{
    if (/^\w+([\.-]?\w+)*@\w+([\.-
    ]?\w+)*(\.\w{2,3})+$/.test(myForm.emailAddr.value))
    {
        return (true)
    }
        alert("You have entered an invalid email address!")
        return (false)
}
```

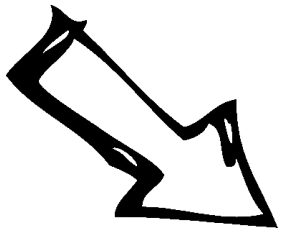
SECURITE AVEC .HTACCESS ET .HTPASSWD

- **.htaccess** : contient l'adresse du fichier **.htpasswd**
- Fichiers cachés avec seulement une extension.

```
AuthName "Page d'administration protégée"
AuthType Basic
AuthUserFile "/home/site/www/admin/.htpasswd"
Require valid-user
```

- **AuthName** : texte qui invitera l'utilisateur à inscrire son login et son mot de passe.
- **AuthUserFile** : chemin absolu vers le fichier **.htpasswd** (à mettre dans le même répertoire que le **.htaccess**).

SECURITE AVEC .HTACCESS ET .HTPASSWD



Pour retrouver son chemin absolu...

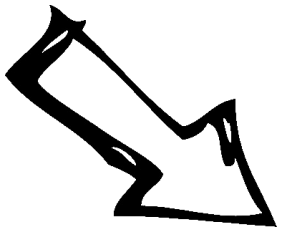
```php
<?php echo realpath('chemin.php'); ?>
```

SECURITE AVEC .HTACCESS ET .HTPASSWD

```
login:mot_de_passe_crypté
```

```
mateo21:$1$MEqT//cb$hAVid.qmmSGFW/wDlIfQ81
ptipilou:$1$/lgP8dYa$sQNXcCP47KhP1sneRIZoO0
djfox:$1$lT7nqnsg$cVtoPfe0IgrjES7Ushmoy.
vincent:$1$h4oVHp3O$X7Ejpn.uuOhJRkT3qnw3i0
```

SECURITE AVEC .HTACCESS ET .HTPASSWD



Crypter un mot de passe.

```php
<?php echo crypt('arcenciel'); ?>
```

BASE DE DONNÉES, ETC...