Drosalys
POUR QUE LE WEB VOUS RÉUSSISSE

INTRODUCTION LANGAGE PHP

Théau Goncalves

Drosalys
POUR QUE LE WEB VOUS RÉUSSISSE

LE LANGAGE PHP

- Rasmus Lerdorf
- 1995 (version 1.0)
- Procédurale
- Typage faible et dynamique
- Interprété

```php
echo "Hello World!!";
```

EXÉCUTER DU CODE PHP
- EN LIGNE DE COMMANDE -

- Prérequis:
  - Avoir l'interpréteur PHP installé

- Ligne de commande (= un environnement d'exécution)
  - Mode interactif

```
php -a
```

  - Exécution de fichiers (php test.php)

```
php test.php
```

STRUCTURE DU CODE PHP

Votre premier fichier de code

Théau Goncalves

STRUCTURE DU CODE PHP

Balise ouvrante
Obligatoire

Instruction PHP
Une instruction se termine par un point virgule

Balise fermante
Optionnelle (inutile s'il n'y a que du code PHP après la balise ouvrante)

```
1. <?php
2.
3.
4. echo "Hello World!!";
5.
6.
7. ?>
```

EXERCICES

- « Hello world! »

1. En utilisant la commande php en mode interactif

2. Créer un fichier et exécuter avec la commande php.

SYNTAXE DU CODE PHP

Théau Goncalves

LES COMMENTAIRES EN PHP

```php
<?php

// Je suis un commentaire en PHP

/*
 * Moi aussi,
 * mais je suis multiligne
 */

```

ASSIGNATION D'UNE VARIABLE EN PHP

```php
<?php

// Assignation d'une variable de type "chaîne de caractères"
$firstName = "Théau";

// Assignation d'une variable de type "entier"
$age = 33;

// Assignation variable de type "booléen"
$isAlive = true;

// Création d'une variable de type "tableau"
$array = [];

// Création d'une variable nulle
$null = null;
```

- Les nom de variable commence par un « $ »
- Déclaration non nécessaire car typage faible

FONCTIONS PHP

```php
<?php


$firstName = "Théau";


// Affiche la longueur d'une chaîne
de caractères

echo strlen($firstName);
```

- Documentation:
  - Recherche via Google
  - Choisir le site php.net

AFFICHER LE CONTENU D'UNE VARIABLE EN PHP

```php
<?php

$firstName = "Théau";

// Affiche une chaîne de caractères
echo $firstName;

// Affiche aussi une chaîne de caractères
print $firstName;

// Affiche le contenu d'une variable sous forme d'une chaîne de caractères
print_r($firstName);


// Affiche les information d'une variable
var_dump($firstName);
```

- Les nom de variable commence par un « $ »
- Déclaration non nécessaire car typage faible

LES OPÉRATEURS ARITHMÉTIQUES

```php
$result = 2 + 5;
$result = 5 - 5;
$result = 5 * 5;
$result = 8 / 2;
$result = 8 % 2;
$result = $result + 5;
$result += 5;
$result *= 2;
$result++;
$result--;
```

- Addition : +
- Soustraction : -
- Multiplication : *
- Division : /
- Modulo : %
- Incrémentation
- Décrémentation

LES OPÉRATEURS DE CHAÎNES

```php
<?php
$hello = "Hello";
$world = "world";


// Concaténation
$result = $hello." ".$world;
// Concaténation aussi
$result .= "!";
echo $result;
```

EXERCICES

- Créer votre « Hello » (*exemple = Hello Theau*) personnalisé en utilisant une variable

1. Tester votre code en ligne de commande

2. Tester votre code via le serveur web

EXÉCUTER DU CODE PHP
- EN PASSANT PAR UN SERVEUR -

- Prérequis:
  - Avoir l'interpréteur PHP installé
  - Avoir un serveur web installé
  - Avoir un navigateur web
- Configurer PHP pour afficher les erreurs
- Configurer l'accès au serveur web
- Configurer le serveur web pour servir les fichiers PHP

EXERCICES

1. A partir d'un âge, on doit indiquer l'année de naissance

2. A partir de ce tableau [12, 15, 19, 2], calculer la moyenne des valeurs.

3. A partir d'un prix HT unitaire d'un produit, ainsi que le nombre de produit. On veut connaitre le total TTC arrondi à 2 chiffres après la virgule.

CONDITIONS

```php
<?php


if ($result >= 50) {
      echo "Your result is at least the average.";
} else if ($result >= 90) {
      echo "Your result is very good.";
 } else {
      echo "Your result is not bad.";
}
```

- Si
- Sinon Si
- Sinon

CONDITIONS : LES OPÉRATEURS

```php
if ($result == 50)
if ($result > 50)
if ($result < 50)
if ($result >= 50)
if ($result <= 50)
if ($result > 80 && $result < 20)
if ($result >= 50 || $result == 0)
if (!$result < 50)
```

- Égale : ==
- Différent !=
- Supérieur : >
- Inférieur : <
- Supérieur ou égale : >=
- Inférieur ou égale : <=

- ET : &&
- OU : ||
- NON : !

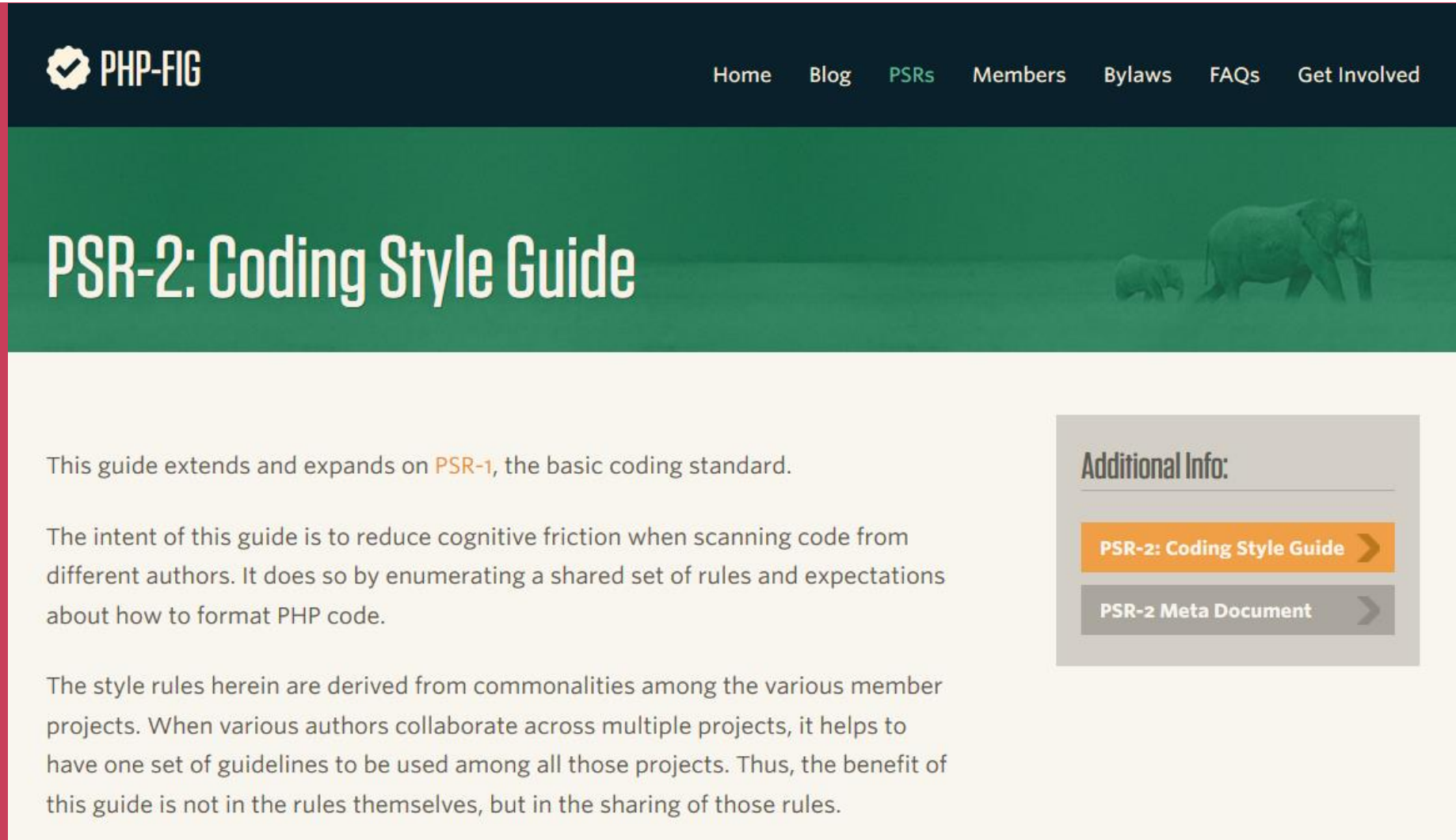
CONDITIONS
- TERNAIRE -

```php
<?php
$result = 52;


if ($result > 50) {
        echo "Your result is above 50!";
} else {
        echo "Your result is under 50...";
}
// Ternaire équivalente à la condition ci-dessus
echo ($result > 50) ? "Your result is above 50!" : "Your result is under 50...";
```

GUIDES POUR LA SYNTAXE DE CODE PHP

- **PSR = PHP Standards Recommendations**
- **Bonnes pratiques**
- http://www.php-fig.org/psr/



EXERCICES

1. On veut à partir d'une moyenne indiquer si la personne à la moyenne ou non. On veut aussi lui préciser sa mention le cas échéant.

2. A partir de la température d'un volume d'eau, on veut savoir dans quel état est l'eau (solide, liquide ou gaz)

DÉCLARER UNE FONCTION

```php
// Déclaration de la fonction "hello"
function hello()
{
    echo "Hello world";
}

// Appel de la fonction "hello"
hello();
```

```
Hello world
```

- Un nom de fonction
- Première lettre en minuscule
- Déclaration avant ou après l'appel

DÉCLARER UNE FONCTION

```php
// Déclaration de la fonction "getHello"
function getHello($name)
{
    return "Hello ".$name;
}

// Appel de la fonction "getHello"
$hello = getHello("Théau");
echo $hello;
```

```
Hello Theau
```

- Peut prendre des arguments
- En générale, retourne une valeur

ARGUMENTS D'UNE FONCTION

```php
// Déclaration de la fonction "getHello"
function getHello($name = "world")
{
    return "Hello ".$name;
}

// Appel de la fonction "getHello"
$hello = getHello();
echo $hello;
```

```
Hello world
```

- On peut définir des valeurs par défaut pour les arguments

ARGUMENTS D'UNE FONCTION

```php
$word = 'test';
// Déclaration de la fonction "toUpper"
function toUpper($word)
{
    $word = strtoupper($word);
}

// Appel de la fonction "toUpper"
toUpper($word);
echo $word;
```

```
test
```

- Par défaut, la fonction n'a pas accès aux variables du code ou est appelé la fonction

ARGUMENTS PAR RÉFÉRENCE

```php
$word = 'test';
// Déclaration de la fonction "toUpper"
function toUpper(&$word)
{
    $word = strtoupper($word);
}

// Appel de la fonction "toUpper"
toUpper($word);
echo $word;
```

```
TEST
```

- Si un argument est passé par référence, la fonction peut le modifier.
- Symbole = « & »

FONCTIONS ANONYMES

```php
$array = ['h', 'e', 'l', 'l', 'o'];

// Anonymous function passed as `array_walk` argument
array_walk(
    // First argument - array
    $array,
    // Second argument - anonymous function
    function ($value) {
        echo $value.PHP_EOL;
    }
)
```

```
h
e
l
l
o
```

- Fonction sans nom…
- Utile dans le cas ou l'appel de la fonction est conditionné par une autre fonction
- Exemple : cas du `array_walk` ou de `spl_autoload_register`

LES TABLEAUX

```php
// Initialisation d'un tableau vide
$array = [];

// Initialise un tableau avec des valeurs
$array = ['test', 15];

// Affiche la valeur à l'index 0 soit test
echo $array[0].PHP_EOL;

// Ajoute une valeur au tableau à l'index suivant, soit l'index 3
$array[] = 12;

// Assigne une valeur à l'index 1 (remplace la valeur 15 par 17)
$array[1] = 17;

// Affiche le tableau
print_r($array);
```

```
test
Array (
        [0] => test
        [1] => 17
        [2] => 12
)
```

- Les éléments peuvent être de types différents
- Index numérique
- Index commence à o

LES TABLEAUX ASSOCIATIFS

```php
// Initialise un tableau avec des valeurs et des clés
$array = [
    'anglais' => 15,
    'math' => 12,
    0 => 13.5
];

// Affiche la valeur pour la clé 'math' soit 12
echo $array['math'].PHP_EOL;

// Ajoute une valeur au tableau à pour la clé 'espagnol'
$array['espagnol'] = 18;

// Affiche le tableau
print_r($array);
```

```
12
Array (
        [anglais] => 15
        [math] => 12
        [0] => 13.5
        [espagnol] => 18
)
```

- On peut définir des clés pour les valeurs d'un tableau
- Les clés sont de type string ou int (= index tableau indicé classique)
- Syntaxe : « clé » => valeur

BOUCLE FOR

```php
for ($i = 1; $i <= 10; $i++) {
    echo $i;
}
```

```
12345678910
```

- Boucle « pour »
- Répétition d'une ou plusieurs opérations.
- Itérations

BOUCLE FOREACH

```php
$array = [
    'math' => 15,
    'english' => 10
];

foreach ($array as $key => $value) {
    echo $key. " : " .$value.PHP_EOL;
}
```

```
math : 15
english : 10
```

- Itérations sur un élément itérable (exemple : tableau)
- La clé et la valeur de chaque itération peut être accessible en déclarant des variables après le mot clé « as »

BOUCLE WHILE

```php
$x = 10;

while ($x < 10) {
    $x++;
}

echo $x;
```

```
10
```

- Boucle « tant que »
- Continue selon un condition
- Attention au boucle infini

BOUCLE DO WHILE

```php
$x = 10;

do {
    $x++;
} while ($x < 10);

echo $x;
```

```
11
```

- Première itération toujours exécutée.
- Utilisée pour « tester » et vérifier une saisie utilisateur.

ARRÊTER UNE BOUCLE

```php
for ($i = 0; $i < 10; $i++) {
    if ($i === 5) {
        break; // arrête la boucle
    }
    echo $i;
}
```

```
01234
```

- Arrêt arbitraire d'une boucle
- Mot clé : break

ARRÊTER UNE ITÉRATION

```php
for ($i = 0; $i < 10; $i++) {
    if ($i === 5) {
        continue; // arrête le tour de boucle actuel (n'arrête pas la boucle)
    }
    echo $i;
}
```

```
012346789
```

- Arrêt arbitraire d'un tour de boucle
- Mot clé : continue

LES OBJET EN PHP

Théau Goncalves

LES CLASSES

Nom de la classe

Attributs de la classe

Constructeur de classe

Méthodes de la classe

```php
class Contact
{
    /** @var string $lastName */
    private $lastName;

    /**
     * Contact constructor.
     * @param $lastName
     */
    public function __construct($lastName)
    {
        $this->lastName = $lastName;
    }

    /**
     * @return string
     */
    public function getLastName()
    {
        return $this->lastName;
    }

    /**
     * @param string $lastName
     */
    public function setLastName($lastName)
    {
        $this->lastName = $lastName;
    }
}
```

LA VISIBILITÉ

- Qui a le droit de « voir » la classe, l'attribut ou la méthode ?

- public
  - Accessible depuis partout, aussi bien la classe courant que les autres.
- protected
  - Accessible uniquement par la classe courant et ses filles (héritage).
- private
  - Accessible uniquement par la classe courant.

UTILISATION DES OBJETS

```php
// Instanciation de la l'objet avec passage d'un argument pour le constructeur
$contact1 = new Contact("Goncalves");

// Appel d'une méthode de l'objet
echo $contact1->getLastName().PHP_EOL;

// Modification d'un attribut en utilisant une méthode
$contact1->setLastName("NOBODY");

echo $contact1->getLastName();
```

```
Goncalves
NOBODY
```

- Instanciation avec le mot clé « new »
- Accès au méthode et au attribut avec la syntaxe : $objet->attribut $objet->methode()
- Attention à la portée des variable (public, private, protected)

ATTRIBUTS ET METHODES « STATIC »

```php
class Contact
{
    public static $type = "Perso";

    public static function getType()
    {
        echo self::$type;
    }
}
```

- Appartiennent à la classe (déclaration avec le mot clé « static »)
- Accès au méthode et au attribut avec la syntaxe : NomDeLaClasse::$attribut NomDeLaClasse::methode()
- Accès possible via une instance: $objet::$attribut $objet::methode()
- Depuis la classe on utilise le mot clé « self »
- Attention à la portée des variable (public, private, protected)

HÉRITAGE ET SURCHARGE DE MÉTHODES

```php
class Employee extends Contact
{
    private $secuNb;

    public function __construct(
        $firstName, $lastName, $secuNb
        )
    {
        parent::__construct($firstName, $lastName);
        $this->secuNb = $secuNb;
    }
}
```

- Mot clé « extends » pour l'héritage
- Accès possible à la classe parente avec le mot clé « parent »
- On peut remplacer une méthode du parent en la redéfinissant chez l'enfant OU la surcharger en appelant la méthode du parent dans la méthode de l'enfant

DÉCLARER UNE INTERFACE

```php
<?php


interface ContactInterface
{
    public function setFirstName($arg);
    public function getFirstName();

    public function setLastName($arg);
    public function getLastName($arg);
}
```

- Mot clé « interface » pour déclarer une interface
- Déclare la portée, le nom et les arguments des méthodes (finit par un « ; »
- L'interface pose uniquement des contraintes de présences, portée et nomenclature de méthode, mais ne précise pas leur fonctionnement

IMPLÉMENTER UNE INTERFACE

```php
<?php

class Contact implements ContactInterface
{
        public function setFirstName($arg)
        {
                $this->firtName = $arg;
        }

        public function getFirstName()
        {
                return $this->firtName;
        }

        public function setLastName($arg)
        {
                $this->lastName = $arg;
        }

        public function getLastName($arg)
        {
                return $this->lastName;
        }
}
```

- Mot clé « implement » pour implémenter une interface
- Une classe implémentant une interface doit déclarer les méthode de l'interfaces et respecter la portée

LES NAMESPACES ET L'AUTOLOAD

Ou comment pallier aux limites d'un include

Théau Goncalves

LES NAMESPACES

- Pour permettre de mieux organiser son code:
  - Un fichier par classe
  - Trier ses classes dans des dossiers et sous dossiers
- Pour pouvoir utiliser des classes ayant le même nom
- Correspond à un enrichissement du nom de la classe
- Voir les PSR-0 et PSR-4 pour la syntaxe
- Exemple de syntaxe:

Namespace	Chemin correspondant
TheauApp\Company\Boss	« racine_du_projet »/classes/Company/Boss.php

LES NAMESPACES

```php
<?php

// Déclaration de l'espace de nom
namespace TheauApp\Company;

// Utilisation du nom de la classe avec namespace pour la classe Contact
class Employee extends \TheauApp\Contact
{
```

- On déclare généralement un espace de nom en début de fichier avec le mot clé « namespace »
- Le namespace enrichis le nom de la classe :
  - La classe « Employee » devient la classe « \RubenApp\Company\Employee »
- Pour désigner un espace de nom absolu (depuis la racine), on commence par un « \ »

LES NAMESPACES

```php
<?php

namespace TheauApp\Company;

// On contextualise l'utilisation de la classe Contact en la déclarant avec son namespace
use TheauApp\Contact;

// le "use" nous permet de noter le nom de la classe Contact sans son namespace
class Boss extends Contact
{
```

- On peut déclarer les noms complet des classes (avec namespace) que l'on va utiliser par la suite, en utilisant le mot clé « use » en début de fichier (!!!après le namespace!!!)
- Tout appel à la classe déclaré avec le « use » sera fait avec le nom complet, sans avoir à le renoter

L'AUTOLOAD

- Fonctionnalité de PHP qui permet d'inclure des fichiers dynamiquement
- On peut enregistrer des fonctions de chargement (exécutant le « include ») = autoload
- Les autoload enregistrés seront déclenchés chaque fois qu'une classe non définie sera appelée dans un fichier
- Les fonction enregistrées en tant qu'autoload avec `spl_autoload_register` auront accès au nom de la classe appelé (avec son namespace) en premier argument.
- Le rôle de l'autoload est de traiter le nom de la classe appelé et d'inclure le fichier correspondant

EXEMPLE D'AUTOLOAD GÉRANT LES NAMESPACES

```php
<?php

// Fonction qui enregistre les autoload (prend une fonction en argument - ici fonction anonyme)
spl_autoload_register(function ($class) { /* $class contient une chaine de caractère correspondant à
la classe appelée
(enrichie de namespace le cas échéant */
    // Tableau contenant les Namespaces root et avec le dossier correspondant en clé
    $namespaces = [
        "classes" => "TheauApp"
    ];
    // On remplace le namespace root par le dossier correspondant
    foreach ($namespaces as $dir => $namespace) {
        if (false !== strpos($class, $namespace)) {
            $class = str_replace($namespace, $dir, $class);
            break;
        }
    }
    // On remplace les \ par le bon séparateur (en fonction de l'OS)
    $class = str_replace('\\', DIRECTORY_SEPARATOR, $class);

    // On complète l'URI relative avec l'extension de fichier php
    $path = $class . '.php';

    // On procède à l'include
    include $path;
});
```

DES QUESTIONS ?

SOURCES

- https://secure.php.net
- http://www.php-fig.org/psr/
- Ruben Martinez