PHP 치트시트

기본 구문

PHP(Hypertext Preprocessor)는 웹 개발에 널리 사용되는 서버 측 스크립트 언어입니다.

- **PHP 코드 블록**: 모든 PHP 코드는 `<?php`로 시작하여 `?>`로 끝나는 태그 내부에 작성됩니다. (단, PHP 코드로만 구성된 파일인 경우 마지막 `?>`는 생략하는 것을 권장합니다)
- **주석**: `//` 또는 `#`을 사용하여 한 줄 주석을 작성하거나, `/* ... */`를 사용하여 여러 줄 주석을 작성합니다.
- **출력**: `echo` 또는 `print`를 사용하여 텍스트를 출력합니다. (일반적으로 `echo`가 미세하게 더 빠르며 널리 사용됩니다)
- **변수**: 변수명은 항상 `$` 기호로 시작하며, 영문자나 밑줄(_)로 시작해야 합니다. (예: `$name = "Jules";`)
- **문자열 연결**: 마침표(`.`) 연산자를 사용하여 문자열을 서로 연결합니다.
- **문자열 보간 (Interpolation)**: 큰따옴표(") 내부에 변수를 직접 사용하여 변수의 값을 문자열에 포함할 수 있습니다.

변수 타입 및 연산자

- **기본 타입**: `String` (문자열), `Integer` (정수), `Float` (실수), `Boolean` (논리형), `Array` (배열), `Object` (객체), `NULL`.
- **배열 (Array)**:
  - 인덱스 배열: 숫자를 키로 사용하는 순차 배열입니다. (예: `$cars = ["Volvo", "BMW"];`)
  - 연관 배열: 문자열을 키로 사용하는 키-값 쌍의 배열입니다. (예: `$ages = ["Peter" ⇒ 35, "Ben" ⇒ 37];`)
- **연산자**: 산술, 할당, 비교, 논리 연산자는 대부분의 다른 프로그래밍 언어와 동일한 방식으로 동작합니다.
- **삼항 연산자**: `(조건) ? 참일_때_값 : 거짓일_때_값;` 형식을 사용하여 조건부 할당을 간결하게 처리합니다.
- **Null 병합 연산자 (PHP 7+)**: `??` 연산자를 사용하여 변수가 설정되어 있지 않거나 `null`인 경우의 기본값을 지정할 수 있습니다.

제어 흐름

- `if-elseif-else`: 조건에 따라 실행 경로를 제어합니다.

```
if ($age < 20) {
  // ...
} elseif ($age < 30) {
  // ...
} else {
  // ...
}
```

- `switch`: 하나의 변수를 여러 값과 비교할 때 사용합니다.
- `while` / `do-while` 루프: 특정 조건이 만족되는 동안 코드를 반복 실행합니다.
- `for` 루프: 정해진 횟수만큼 반복할 때 유용합니다.
- `foreach` 루프: 배열이나 객체의 모든 요소를 순차적으로 순회할 때 가장 효율적인 방식입니다.

```
$colors = ["red", "green", "blue"];
foreach ($colors as $color) {
  echo "색상: $color\n";
}
```

함수

- **함수 정의**: `function` 키워드를 사용하며, 기본 매개변수 값을 지정할 수 있습니다.

```
function greet($name = "Guest") {
  return "안녕하세요, $name님!";
}
```

- **타입 힌팅 (Type Hinting)**: 매개변수와 반환값의 타입을 명시하여 코드의 안정성을 높입니다. (PHP 7 이상 권장)

```
function add(int $a, int $b): int {
  return $a + b;
}
```

클래스와 객체 (OOP)

- **클래스 정의**: 객체의 속성(Property)과 동작(Method)을 정의합니다.

```
class Car {
  public $color;

  // 생성자 (PHP 8 이상에서는 속성 승격 기능을 지원합니다)
  public function __construct(public string $model) {}

  public function drive() {
    return "$this→model 차량이 주행 중입니다.";
  }
}
```

- **객체 생성 및 접근**: `new` 키워드로 인스턴스를 생성하고 `→` 연산자로 멤버에 접근합니다.
- **상속**: `extends` 키워드를 사용하여 부모 클래스의 기능을 물려받습니다.
- **인터페이스 & 트레이트**: 인터페이스(`interface`)로 명세를 정의하거나, 트레이트(`trait`)로 코드를 재사용합니다.

슈퍼글로벌 (Superglobals)

스크립트 내 어느 위치에서든 별도의 선언 없이 즉시 접근할 수 있는 미리 정의된 내장 변수들입니다.

- `$_GET`: URL 쿼리 스트링을 통해 전달된 데이터.
- `$_POST`: HTTP POST 요청 바디를 통해 전달된 데이터.
- `$_REQUEST`: `$_GET`, `$_POST`, `$_COOKIE`의 합집합.
- `$_SERVER`: 서버 환경 및 실행 헤더 정보.
- `$_SESSION`: 서버 측 세션 데이터 관리.
- `$_COOKIE`: 브라우저에 저장된 쿠키 값.
- `$_FILES`: HTTP POST 방식으로 업로드된 파일 정보.

파일 처리

- **파일 읽기**: `file_get_contents()`를 사용하여 파일 전체 내용을 문자열로 읽어옵니다.
- **파일 쓰기**: `file_put_contents()`를 사용하여 문자열을 파일에 저장합니다.

데이터베이스 연동 (PDO)

PHP Data Objects (PDO)는 서로 다른 다양한 데이터베이스 엔진에 대해 일관된 방식으로 접근하고 조작할 수 있도록 설계된 인터페이스입니다.

```
// 데이터베이스 연결 설정 예시
$dsn =
"mysql:host=localhost;dbname=test;charset=utf8mb4";
$options = [
    PDO::ATTR_ERRMODE ⇒
PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE ⇒
PDO::FETCH_ASSOC,
];

try {
     $pdo = new PDO($dsn, $user, $pass,
$options);
} catch (\PDOException $e) {
     // 연결 에러 발생 시 처리
}

// Prepared Statement를 사용한 안전한 쿼리 실행 (SQL Injection 방지)
$stmt = $pdo→prepare('SELECT * FROM users WHERE email = ?');
$stmt→execute([$email]);
$user = $stmt→fetch();
```

Composer (의존성 관리)

PHP 생태계의 표준 패키지 관리 도구입니다.

- `composer.json`: 프로젝트가 의존하는 라이브러리 목록을 관리합니다.
- `composer install`: 정의된 의존 라이브러리들을 설치합니다.
- `composer update`: 라이브러리들을 허용된 범위 내의 최신 버전으로 업데이트합니다.
- `composer require`: 새로운 패키지를 프로젝트에 추가합니다.
- **Autoloading**: `require 'vendor/autoload.php';` 한 줄로 모든 설치된 클래스를 자동으로 로드할 수 있습니다.