Aula 1: Evolução do PHP

* PHP 1.0(1995)
  + Personal Home Page;
  + Funcionalidades simples para um site básico;
  + Não tinha características de linguagem em si.
* PHP 2.0(1997)
  + Oficialmente nomeado PHP/FI 2.0;
  + Quase um Linguagem de programação;
  + Recursos limitados.
* PHP 3.0(1998)
  + Primeira versão colaborativa;
  + Características mais relacionadas a uma linguagem;
  + Extension API.
* PHP 4.0(2000)
  + Zend Engine 1.0;
  + Melhoria de Performance;
  + Implementação de super globais( $\_*GET, $\_*POST, $\_SESSION…);
  + Aumento da modularização.
* PHP 5.0(2004)
  + Zend Engine 2.0;
  + Primeira versão com Orientação a Objetos;
  + PHP Data Objects(PDO)
  + Operador de Exponenciação(!, ??, !!, ::)
  + Suporte a JSON
  + Namespaces(tipo de encapsulamento), closures, garbage collection(descarte de dados não necessários), exception( throw new Exception()...) e handling;
  + Novo depurador (phpdbg).
* PHP “6.0”(2005)
  + Versão sonhada somente pela comunidade;
  + Nunca recebeu uma release oficial;
  + Prometia aumento de segurança;
  + Suporte Nativo a Unicode;
* PHP 7.0(2015)
  + Zend Engine 3.0;
  + Performance até 9x maior;
  + Escape de código Unicode;
  + Operador null coalescing (??);
  + Declaração escalar;
  + Operador spaceship <=>;
  + Classes Anônimas;
  + Exceções mais modernas;
  + Retorno de void;
  + Libsodium(biblioteca de criptografia de dados)
  + Foreign Functions interface(FFI), ou seja, habilita o uso de bibliotecas de outras linguagens de programação, tendo uma maior liberdade de funcionalidades ao PHP.
* PHP 8.0(2020)
  + Zend Engine 4.0
  + Just-In-Time compilation(Compilador que ao ver scripts mais pesados, ao invés de fazer a máquina virtual da Zend(ZVM) para compilar o programa, ele deixa a própria máquina do código(ou seja o servidor local) para poder aumentar sua performance, porém isso acontece somente com códigos pesados, além disso todos os códigos serão compilados pela máquina virtual da Zend.
  + Locale-independent number conversion;
  + Named arguments;
  + Expressão match;
  + Operador nullsafe(Melhor tratamentos de nulos[?]);
  + Enumerations(Variáveis que aceitam valores pré-definidos);
  + Readonly properties(bom para o POO);
  + Fibers.

Modularização: Modularização é um princípio de design de software que consiste em dividir um programa grande em módulos menores e independentes. Isso é feito para tornar o código mais organizado, fácil de manter e reutilizável. Cada módulo é uma unidade de código que executa uma tarefa específica e pode ser desenvolvido, testado e mantido separadamente dos outros módulos.

No contexto do PHP, a modularização pode ser alcançada por meio de arquivos individuais que contêm funções ou classes relacionadas.