Основы php.

Основное отличие php от js:

1. Способ наименования переменных, с символа $, после него всё, кроме цифры.

$\_1 = 0; //правильно

$1\_ = 0; //неправильно

1. Очень требователен к «;».
2. В php нет типа данных undefined.
3. Конкатенация строк осуществляется с помощью инструкции «.».

$a = ‘Вася’;

$b = ‘Миша’;

$a.$b

1. В php можно вставить значение в строку «из коробки».
2. Доступ до методов свойств экземпляров осуществляется стрелочкой «->».
3. Массивы в php можно объявить через квадратные скобки.

$arr = [];

1. Добавление в массив новых элементов:

$arr[] = 123;

1. Ассоциативные массивы:

$arr = array(‘name ’ => 9,

‘age’ => 8);

$arr[‘name’] = 123;

1. Объект, реализован как экземпляр класса std:

$obj = new stdClass();

$obj->key1=’Вася’;

$obj->key2=18’;

1. Сильное достоинство php в том, что там реализован по ООП и реализованы нормальные классы.

class Point {}

Наследование классов с помощью команды extends.

1. Модификаторы доступа:

class Name{

public

protected

private

}

// модификаторы могут быть заданы для любого элемента, по стандарту стоит модификатор public.

Public – позволяет обратиться к методу или свойству класса в экземпляре класса, то есть снаружи, и в классе потомке к методу или свойству родителя.

Protected – запрещает вызов или обращение к свойству в экземпляре класса, но можно вызывать родительские методы в классе потомке.

Private – это protected, только нельзя вызывать методов классов родителей в классе потомке.

1. У классов в php есть метод конструктора и метод деструктора:

class Name{

function \_\_construct() {}

function \_\_destruct() {}

}

Они пишутся без модификатора доступа, потому что они уже имеют доступ public и его нельзя изменить.

Вызов конструктора родителя:

Parent::\_\_construct();

1. В php можно управлять вызовом конструктора родителя, потому что вызов конструктора родителя является необязательным. Конструктор родителя можно явно вызвать в конструкторе потомка, в случае его не вызова, он не будет вызван. Если в родителе конструктор есть, а в потомке нету, то в потомке конструктор родителя будет вызван автоматически.
2. В php есть понятие статичного класса – это класс методы, которого могут быть вызваны, без создания метода экземпляра класса.
3. Вызов метода или свойства из контекста:

$this->a

Note: $ - не является составной частью чего либо, это спецсимвол указывающий на то, что это после него идет имя переменной.

Супер глобальные переменные, их можно использовать из любой части программы:

* $\_GET – ассоциативный массив, в котором лежат все входные параметры.
* $\_POST – тоже ассоциативный массив, в котором лежат входные данные из post запроса.

Note: основной принцип работы php в том, её код срабатывает только тогда, когда на сервер поступает команда об отработке серверной части приложения.

Варианты запроса в бэкеднд: get, post, insert, delete и тд.

У Get запроса отсутствует тело запроса и параметры запроса задаются в адресной строке.

У post запроса есть тело(body) и поэтому параметры пост запроса можно не только в адресной строке, но и его теле.

Принципиальная разница между get и post запроса в том, что get можно отправить из адресной строки браузера, то есть ему не нужен клиент для отправки, а post запрос не отправить без задавания его параметров в теле.

Также get запрос имеет ограничение на длину и количество символов, и для большого количества переменных более удобен post запрос.

Мы в наших проектах в основном будем использовать get запросы.

В запросах могут быть заголовки и к ним могут быть прикреплены куки файлы.

Основной способ обеспечения аутентификации пользователя отправляющего запрос в бэкенд – это определение пользователя по специализированному ключу(токену), который имеет ограниченный срок жизни, и выдаётся клиенту в рамках сессии.

Данные в бэкенде хранятся в базе данных.

Работа приложения:

Клиент(html, css, js) <=> сервер(php) <=> бд

Для снижения нагрузки на бэкенд применяются механизм кэширования, суть которого в том, что каждый новый запрос в бэкенд после его обработки формируется ответ и он складывается в кэш, который имеет ограниченный срок жизни и в случае аналогичного запроса, бэкенд отвечает на него ответом из кэша. Правильно настроенный кэш позволяет сократить нагрузку на 90%.

Минус кэширования в том, что данные могут быть не актуальные для пользователя, также пользователи видят одну и ту же информацию.

Принцип работы бэкенда на проекте(личное мнение препода): Любой запрос на бэкенд должен быть обработан единообразно и должен быть дан стандартный ответ. В рамках нашей работы любой ответ с бэкенда, является стандартным.

Стандартный ответ:

* имеет формат json, его структура:
  + result: ‘ok’|’error’;
  + data: any;
  + error?: {code: number, // код ошибки

text: string };// текст ошибки

В проекте должен быть написан документ с названием «API». В нем должны быть описаны абсолютно все запросы с которыми может работать бэкенд, а именно параметры запроса, вид ответа, а также возможные ошибки. Ответственным за него являются бэкендеры.

Также если необходимо написать новый метод на бэкенде, сначала он описывается в файле API.md: адрес, формат, ошибки, аннотации и только потом идет работа по программированию метода.

Дезешечка: на основании имеющегося проекта, необходимо разработать документацию по методам необходимым в рамкам проекта( буквально API.md), также можно сходить к Aлексею Сергеевичу за примером.

На клиенте необходимо разработать структуру работы клиента и описать необходимый функционал.

Если клиент надо разрабатывать, но бэкенд ещё не сделан, то есть такое понятие «замокать» данные, то есть подменить данные с бэкенда в клиенте подходящими.