1. Описать историю создания языка HTML.
2. Перечислить принципы гипертекстовой разметки.

Принципы гипертекстовой разметки включают:

* Гипертекст: Возможность создания текста с гиперссылками, позволяющими переходить к другим документам или разделам.
* Разметка: Использование тегов для определения структуры и оформления содержимого.
* Интерактивность: Возможность взаимодействия пользователя с документом, например, заполнение форм или выполнение действий при нажатии на элементы.
* Многоуровневость: Возможность создания структурированных документов с разными уровнями вложенности.

1. Описать основные принципы синтаксиса языка гипертекстовой разметки.
2. Раскрыть структуру HTML-документа.
3. Описать физическое и логическое форматирование текста.
4. Объяснить абсолютные и относительные ссылки.
5. Изложить особенности вставки графики, звука и видео на WEB-страницу.
6. Описать создание списков: нумерованного, ненумерованного, списка определений.
7. Описать создание ссылок на документы и файлы.

Нужно использовать тег <a>, атрибут href для указания адреса ссылки и внутренний текст тега для отображения пользователю. Давайте подробно рассмотрим:

Ссылка на веб-страницу:

* <a href="http://www.example.com">Ссылка на веб-страницу</a>

Ссылка на локальный файл в том же каталоге:

* <a href="filename.html">Ссылка на локальный файл</a>

Ссылка на локальный файл в другом каталоге:

* <a href="path/to/filename.html">Ссылка на локальный файл в другом каталоге</a>

Ссылка на скачивание файла:

* <a href="path/to/file.pdf" download>Ссылка на скачивание файла</a>

Ссылка на отправку электронной почты:

* <a href="mailto:info@example.com">Ссылка на отправку электронной почты</a>

Ссылка на вызов JavaScript функции:

* <a href="javascript:myFunction()">Ссылка на JavaScript функцию</a>

1. Описать понятие и назначение изображений-карт (карт-ссылок).
2. Изложить случаи размещения изображений-карт на веб-страницах.
3. Описать создание таблиц в языке HTML и работу с ними.
4. Раскрыть синтаксис составления таблицы средствами HTML.
5. Описать создание форм в языке HTML.

Создание форм в HTML позволяет пользователям взаимодействовать с вашим веб-сайтом, отправлять данные на сервер для обработки и получать обратную связь. Для создания форм используется тег <form>, а также различные элементы управления, такие как <input>, <textarea>, <select> и другие. Давайте рассмотрим основные шаги по созданию формы в HTML подробнее:

* Тег <form>:

<form action="/submit-form.php" method="post">

<!-- Ваши элементы управления будут здесь -->

</form>

* action - атрибут, указывающий URL, на который будут отправлены данные формы для обработки.
* method - атрибут, определяющий метод отправки данных формы (обычно "get" или "post").
* Элементы управления:

Input (ввод данных):

<input type="text" name="username" placeholder="Введите имя пользователя">

* type - тип поля ввода данных.
* name - имя поля, которое будет отправлено на сервер.
* placeholder - текстовая подсказка для пользователя.

Textarea (многострочное текстовое поле):

<textarea name="message" rows="4" cols="50"></textarea>

* name - имя поля.
* rows и cols - количество строк и столбцов текста соответственно.

Select (выпадающий список):

<select name="gender">

<option value="male">Мужчина</option>

<option value="female">Женщина</option>

</select>

<select name="gender">

<option value="male">Мужчина</option>

<option value="female">Женщина</option>

</select>

* name - имя поля.
* <option> - опции выбора внутри списка.

Checkbox (флажок):

<input type="checkbox" name="subscribe" value="yes">

* type - тип поля.
* name - имя поля.
* value - значение, которое будет отправлено на сервер, если флажок отмечен.

Radio (переключатель):

<input type="radio" name="payment" value="credit\_card">

<input type="radio" name="payment" value="paypal">

* type - тип поля.
* name - имя группы радиокнопок. Позволяет выбрать только один вариант.
* value - значение выбранной радиокнопки.
* Кнопка отправки:

<input type="submit" value="Отправить">

* type - тип кнопки (submit для отправки формы).

Пример формы целиком:

<form action="/submit-form.php" method="post">

<input type="text" name="username" placeholder="Введите имя пользователя"><br>

<textarea name="message" rows="4" cols="50"></textarea><br>

<select name="gender">

<option value="male">Мужчина</option>

<option value="female">Женщина</option>

</select><br>

<input type="checkbox" name="subscribe" value="yes"> Подписаться на рассылку<br>

<input type="radio" name="payment" value="credit\_card"> Кредитная карта<br>

<input type="radio" name="payment" value="paypal"> PayPal<br>

<input type="submit" value="Отправить">

</form>

1. Описать теги создания форм и элементов управления, используемых в формах.
2. Охарактеризовать понятие и назначение скриптового языка CSS.
3. Описать преимущества языка CSS.
4. Раскрыть способы связывания таблиц стилей с html-документом.
5. Описать селекторы в языке CSS.
6. Изложить понятие о динамическом HTML.
7. Охарактеризовать объектную модель документа.
8. Охарактеризовать добавления сценария, написанного на языке JavaScript на HTML-страницу.
9. Описать и прокомментировать структуру кода языка JavaScript.
10. Описать лексическую структуру языка JavaScript.
11. Объяснить, что представляют собой клиентские сценарии языка JavaScript.
12. Изложить иерархию классов языка JavaScript.
13. Охарактеризовать объект Array языка JavaScript.
14. Описать объект Data языка Java Script.
15. Объяснить циклы в языке Java Script.
16. Представить и объяснить синтаксис пользовательских функций языка JavaScript.
17. Продемонстрировать и прокомментировать общий вид синтаксиса while или do…while в языке Java Script.
18. Охарактеризовать оператор выбора SWITCH в языке JavaScript.
19. Охарактеризовать методы библиотеки JQuery которые позволяют манипулировать вставляемыми элементами (after(), befor(), insertAfter(), insertBefor()).
20. Изложить назначение и возможности библиотеки JQuery.
21. Описать подключение библиотеки JQuery.
22. Расписать синтаксис JQuery -команд.
23. Охарактеризовать селекторы библиотеки JQuery.
24. Описать использование фильтров odd, eq(), gt(), It() в библиотеке jquery.
25. Перечислить возможные селекторы форм библиотеки JQuery.
26. Изложить основные методы создания анимационных эффектов и синтаксис их использования в библиотеке JQuery.
27. Охарактеризовать библиотеку JQuery.
28. Раскрыть назначение фильтров атрибут библиотеки JQuery.
29. Изложить сущность технологии XML.
30. Охарактеризовать назначение технологии XML.
31. Изложить структуру XML-документа.
32. Описать пролог, тело документа (элементы, атрибуты, комментарии инструкции обработки, секции CDATA).
33. Охарактеризовать пространство имен XML-документа.
34. Сформулировать технологии обработки данных в формате XML.
35. Описать схему DTD, недостатки технологии XML DTD перед XML Schema.
36. Описать назначение веб-сервера Apache, его возможности.
37. Описать возможности языка PHP.
38. Охарактеризовать внедрение PHP-сценариев в HTML-документ.
39. Описать особенности синтаксиса языка PHP.
40. Перечислить типы данных в PHP.
41. Описать способ возвращения ссылки на переменную в языке PHP
42. Написать и прокомментировать синтаксис создания стандартных функций.
43. Описать и прокомментировать форматы функции $() в языке PHP
44. Охарактеризовать функции. slideDown([Duration], [easing], [callback]). slideUp([duration]], [easing], [callback]) библиотеки JQuery.
45. Объяснить стандартные функции языка PHP для управления временем.

PHP предоставляет множество встроенных функций для управления временем, которые позволяют получать текущее время, форматировать даты и времена, выполнять операции с временными интервалами и многое другое. Давайте рассмотрим некоторые из стандартных функций языка PHP для управления временем:

* date(): Эта функция используется для форматирования текущей даты и времени по заданному шаблону.

Пример:

echo date("Y-m-d H:i:s"); // Выводит текущую дату и время в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС

* time(): Функция возвращает количество секунд, прошедших с начала эпохи Unix (1 января 1970 года 00:00:00 UTC).

Пример:

echo time(); // Выводит текущее время в секундах с начала эпохи Unix

* strtotime(): Позволяет преобразовать строку с датой и временем в число секунд, прошедших с начала эпохи Unix.

Пример:

echo strtotime("next Monday"); // Возвращает количество секунд до следующего понедельника

* mktime(): Создает метку времени на основе заданных аргументов (часы, минуты, секунды, месяц, день, год).

Пример:

echo mktime(12, 30, 0, 3, 15, 2024); // Создает метку времени для 15 марта 2024 года в 12:30

* gmdate(): Аналогична функции date(), но выводит дату и время в формате GMT (времени по Гринвичу).

Пример:

echo gmdate("Y-m-d H:i:s"); // Выводит текущую дату и время в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС по Гринвичу

* strftime(): Функция форматирует дату и время с учетом текущей локали.

Пример:

setlocale(LC\_TIME, "ru\_RU.utf8"); // Установка русской локали

echo strftime("%A, %d %B %Y, %H:%M:%S"); // Выводит текущую дату и время на русском языке

1. Написать и прокомментировать функции языка PHP для управления временем.
2. Описать создание функций, определяемых пользователем в языке PHP.

Создание пользовательских функций в PHP позволяет вам определять собственные блоки кода, которые могут быть повторно использованы в вашем приложении. Это удобно для упрощения кода, избегания дублирования и улучшения его читаемости. Давайте рассмотрим основы создания пользовательских функций в PHP:

// Синтаксис создания пользовательской функции

function functionName($param1, $param2, ...) {

// тело функции

// код, который нужно выполнить

return $result; // необязательное возвращаемое значение

}

* function: это ключевое слово, с помощью которого определяется функция.
* functionName: это имя вашей функции. Выбирайте его так, чтобы оно было описательным и легко запоминалось.
* $param1, $param2, ...: это параметры функции, которые она принимает. Они могут быть любыми скалярными типами данных (числа, строки, логические значения) или массивами.
* return: это ключевое слово, которое указывает, что функция возвращает значение. Возвращаемое значение может быть любого типа данных.

Пример создания пользовательской функции в PHP:

// Функция для вычисления суммы двух чисел

function add($a, $b) {

$sum = $a + $b;

return $sum;

}

// Вызов функции и вывод результата

$result = add(5, 3);

echo "Сумма чисел: " . $result; // Выведет "Сумма чисел: 8"

1. Написать и прокомментировать синтаксис функций, определяемых пользователем.

В PHP синтаксис функций, определяемых пользователем, довольно прост и похож на синтаксис функций в других языках программирования. Давайте напишем простую функцию и прокомментируем её синтаксис:

// Определение функции для вычисления квадрата числа

function square($number) {

// Тело функции

$result = $number \* $number; // Возводим число в квадрат

return $result; // Возвращаем результат

}

// Вызов функции и вывод результата

$number = 5;

$squaredNumber = square($number); // Вызываем функцию square с аргументом $number

echo "Квадрат числа $number равен $squaredNumber"; // Выводим результат

Теперь прокомментируем каждую строку кода:

* function square($number) {: здесь мы начинаем определение функции square. Ключевое слово function используется для создания новой функции. Имя функции - square. В скобках указываются параметры функции, в данном случае это $number, то есть число, для которого мы будем вычислять квадрат.
* $result = $number \* $number;: это тело функции. Здесь мы выполняем вычисление квадрата числа, переданного в параметре $number, и сохраняем результат в переменную $result.
* return $result;: возвращаем результат выполнения функции. Ключевое слово return используется для возврата значения из функции. В данном случае мы возвращаем квадрат числа.
* $number = 5;: здесь мы устанавливаем значение переменной $number равным 5.
* $squaredNumber = square($number);: вызываем функцию square с аргументом $number и сохраняем результат в переменной $squaredNumber.
* echo "Квадрат числа $number равен $squaredNumber";: выводим результат выполнения функции на экран. Мы используем переменные $number и $squaredNumber внутри строки с помощью интерполяции строк, чтобы вывести значение переменных.

1. Перечислить категории аргументов функций в языке PHP.

В языке PHP аргументы функций могут быть разделены на несколько категорий в зависимости от их особенностей и специфики использования. Вот некоторые из основных категорий аргументов функций в PHP:

* Обязательные аргументы: это аргументы, которые должны быть переданы функции при её вызове. Если обязательные аргументы не будут переданы, PHP выдаст ошибку.

Например:

function greet($name) {

echo "Привет, $name!";

}

greet("Мария"); // Передаем обязательный аргумент $name

* Необязательные аргументы с значениями по умолчанию: это аргументы, для которых предоставлено значение по умолчанию. Если значение не передается при вызове функции, используется значение по умолчанию.

Например:

function greet($name = "Гость") {

echo "Привет, $name!";

}

greet(); // Вызов функции без передачи аргумента

* Аргументы переменной длины (аргументы-массивы): это аргументы, которые могут принимать переменное количество значений. Они передаются в виде массива. В PHP их можно использовать с помощью функций func\_get\_args(), func\_get\_arg(), func\_num\_args() или оператора ... (с PHP 5.6+).

Например:

function sum(...$numbers) {

$total = 0;

foreach ($numbers as $number) {

$total += $number;

}

return $total;

}

echo sum(1, 2, 3, 4); // Выведет 10

* Аргументы по ссылке: это аргументы, которые передаются в функцию по ссылке, а не по значению. Изменения, внесенные в такие аргументы внутри функции, отражаются на исходных переменных за пределами функции. Это достигается с помощью символа амперсанда & перед именем аргумента в сигнатуре функции.

Например:

function increment(&$number) {

$number++;

}

$x = 5;

increment($x);

echo $x; // Выведет 6

1. Описать использование глобальных переменных в языке PHP.

В PHP глобальные переменные — это переменные, которые могут быть доступны из любой части вашего кода, включая внутри функций, классов и методов, даже если они объявлены за пределами этих областей. Использование глобальных переменных может быть удобным, но также может привести к путанице и проблемам с областью видимости. Давайте рассмотрим основные способы использования глобальных переменных в PHP:

* Объявление глобальной переменной:

$globalVariable = 10;

Переменная $globalVariable теперь доступна в любом месте вашего скрипта.

* Использование глобальных переменных внутри функций:

function myFunction() {

global $globalVariable;

echo $globalVariable;

}

myFunction(); // Выведет значение переменной $globalVariable

* Изменение глобальной переменной внутри функции:

function updateGlobalVariable() {

global $globalVariable;

$globalVariable = 20;

}

updateGlobalVariable();

echo $globalVariable; // Выведет значение 20

* Использование массива $GLOBALS:

В PHP есть глобальный массив $GLOBALS, который содержит все глобальные переменные в виде ключей массива. Используя этот массив, вы можете получить доступ к глобальным переменным без использования ключевого слова global.

echo $GLOBALS['globalVariable']; // Выведет значение переменной $globalVariable

Использование глобальных переменных в классах:

Для доступа к глобальным переменным внутри класса необходимо также использовать ключевое слово global или массив $GLOBALS.

1. Охарактеризовать назначение аргументов функций в языке PHP.

Аргументы функций в PHP играют важную роль и предоставляют множество возможностей для создания гибкого и мощного кода. Они используются для передачи данных внутрь функций и предоставляют возможность работы с этими данными внутри функций. Вот основные назначения аргументов функций в PHP:

* Передача данных: Основная цель аргументов функций - передача данных внутрь функций для их дальнейшей обработки. Это позволяет функциям работать с различными данными и выполнять различные операции в зависимости от переданных значений.
* Пользовательская настройка функций: Аргументы функций могут использоваться для настройки поведения функций в соответствии с потребностями пользователя. Например, функция для вычисления суммы двух чисел может принимать эти числа в качестве аргументов, что позволяет пользователям передавать различные числа для вычисления.
* Управление состоянием функций: Аргументы функций могут служить для управления состоянием функций, включая включение или выключение определенных функциональностей, передачу дополнительной информации или управление внутренними переменными функций.
* Повторное использование кода: Использование аргументов позволяет создавать универсальные функции, которые могут быть использованы в различных контекстах и с различными данными. Это способствует повторному использованию кода и уменьшает дублирование.
* Улучшение читаемости и поддерживаемости кода: Правильное использование аргументов функций делает код более читаемым и легким для понимания. Путем передачи данных через аргументы функции становятся более независимыми и могут быть легко изменены или модифицированы в будущем.
* Управление областью видимости переменных: Аргументы функций могут использоваться для передачи переменных внутрь функции с целью изменения их значений или использования внутри функции без создания глобальных переменных.

1. Прокомментировать использование статических переменных в языке PHP.

Статические переменные в PHP представляют собой переменные, которые сохраняют свое значение между вызовами функции, но существуют только в области видимости этой функции. Они инициализируются только один раз во время выполнения программы и сохраняют свое значение в течение всего времени выполнения программы. Применение статических переменных имеет ряд особенностей и преимуществ:

* Сохранение состояния между вызовами функции: Статические переменные позволяют функции сохранять свое состояние между вызовами. Например, если функция должна считать, сколько раз она была вызвана, статическая переменная может использоваться для этой цели.
* Локализация переменной в области видимости функции: Статические переменные ограничены областью видимости функции, в которой они определены. Это помогает избежать конфликтов имен и делает код более структурированным.
* Экономия ресурсов: поскольку статические переменные сохраняют свое значение между вызовами функции, это может сэкономить ресурсы, которые были бы затрачены на повторную инициализацию переменной при каждом вызове функции.
* Использование в рекурсивных функциях: Статические переменные могут быть полезными в рекурсивных функциях, где сохранение состояния между итерациями необходимо для корректной работы.

Пример использования статической переменной в PHP:

function countCalls() {

static $count = 0; // Инициализация статической переменной

$count++; // Увеличение счетчика вызовов

echo "Функция была вызвана $count раз(а).<br>";

}

countCalls(); // Выведет "Функция была вызвана 1 раз."

countCalls(); // Выведет "Функция была вызвана 2 раз."

countCalls(); // Выведет "Функция была вызвана 3 раз."

1. Охарактеризовать возвращаемые значения функций языка PHP.

В PHP функции могут возвращать значения с помощью ключевого слова return. Возвращаемые значения функций могут быть любого типа данных: скалярными (целые числа, строки, логические значения), массивами, объектами и т. д. Особенности возвращаемых значений функций в PHP включают в себя следующее:

* Одиночное возвращаемое значение: Функции в PHP могут вернуть только одно значение. Однако это значение может быть комплексным, например, массивом или объектом.

Пример:

function add($a, $b) {

return $a + $b;

}

$result = add(3, 4); // $result будет равен 7

* Любой тип данных: Функции в PHP могут возвращать значения любого типа данных, включая скалярные типы (числа, строки, логические значения), массивы, объекты и null.

Пример:

function getArray() {

return [1, 2, 3];

}

$array = getArray(); // $array будет массивом [1, 2, 3]

* NULL как возвращаемое значение по умолчанию: Если в функции не указан оператор return, она автоматически возвращает значение null. Это может быть полезно, если функция не должна возвращать какие-либо данные.

Пример:

function doSomething() {

// Нет оператора return

}

$result = doSomething(); // $result будет равен null

* Использование в выражениях: Возвращаемые значения функций могут использоваться непосредственно в выражениях, присваиваниях и других операциях.

Пример:

function multiply($a, $b) {

return $a \* $b;

}

$result = multiply(2, multiply(3, 4)); // $result будет равен 24

* Использование ключевого слова return без значения: В функциях можно использовать оператор return без значения для немедленного завершения выполнения функции.

Пример:

function validate($input) {

if (empty($input)) {

return; // Немедленный выход из функции

}

// Другие проверки...

}

1. Написать и прокомментировать синтаксис возвращаемого значения языка PHP.

Синтаксис возвращаемого значения в PHP прост и понятен. В PHP возвращаемое значение определяется с помощью ключевого слова return. Вот пример простой функции с возвращаемым значением и комментариями к синтаксису:

function add($a, $b) {

$sum = $a + $b; // Вычисление суммы аргументов

return $sum; // Возвращаем результат вычисления

}

* return: это ключевое слово, которое указывает на возвращаемое значение функции.
* $sum: это переменная, в которой хранится результат операции сложения аргументов функции.
* return $sum;: эта строка указывает, что функция должна вернуть значение, хранящееся в переменной $sum.

Пример использования функции:

$result = add(3, 4); // Вызываем функцию с аргументами 3 и 4

echo $result; // Выводим результат на экран (7)

Возвращаемые значения в PHP могут быть любого типа данных, включая целые числа, строки, массивы, объекты и т. д. Это позволяет функциям возвращать различные результаты в зависимости от потребностей вашего кода.