**Билет 1. Назначение языка JavaScript. Способы размещения сценариев JavaScript. Структура кода.**

С помощью языка JavaScript можно создавать интерактивные современные веб – приложения. JavaScript полностью интегрируется с HTML/CSS, поддерживается всеми основными браузерами.

**JavaScript** – язык сценариев (скриптов), которые представляют собой программный код не требующий предварительной обработки (компиляции) перед запуском. Код интерпретируется браузером во время загрузки веб - страницы и выполняется.

Кроме размещения текста сценария в теле HTML – документа его можно поместить в отдельный файл. Это должен быть отдельный текстовый файл, с расширением **\*.js**. В этом файле не должно быть ничего кроме операторов JavaScript. В тексте HTML - документа содержимое контейнера <script> приведите к следующему виду:

<script src=”\*.js”></script >

Под размещением сценария понимается расположение контейнера <script> в документе. Этот контейнер может быть расположен как в заголовке документа (между тегами <head>), так и в теле документа, (между тегами <body>). Один документ может содержать произвольное количество сценариев, причем они не будут изолированы друг от друга. Все переменные, функции и объекты, созданные в одном из контейнеров <script>, будут доступны и во всех остальных сценариях, расположенных далее в документе.

Любой код состоит из инструкций. Инструкция – это синтаксические конструкции и команды, которые выполняют определенные действия. В примере была использована конструкция alert ("Добро пожаловать!"), которая отображает сообщение «Добро пожаловать!».

В коде может быть любое количество инструкций, каждая из которых отделяется точкой с запятой.

**Билет 2. Переменные в JavaScript. Требования к переменным**

Переменные используются для хранения информации (данных).

Для создания переменных используется ключевое слово **let**.

let message;

Далее, через оператор присваивания (=), заносим в нее данные:

let message = ‘Hello’;

Строка сохраняется в области памяти, связанной с переменной, и мы можем получить к ней доступ, используя имя переменной:

let message = ‘Hello’;

alert(message);

**Требования к именам переменных:**

1. Имя переменной должно содержать только буквы, цифры и символы $ и \_.
2. Первый символ не должен быть цифрой.
3. Имя переменной может быть любой длины;
4. Имена переменных чувствительны к регистру символов;
5. В качестве имен нельзя использовать зарезервированные слова JavaScript

**ПЕРЕМЕННУЮ МОЖНО ОБЬЯВИТЬ ТОЛЬКО ОДИН РАЗ**

Билет 3. Типы данных в JS.Функции преобразования типов данных.

Переменная JavaScript может содержать любые данные. Язык относится к **динамически типизированным**, т.е. переменные не привязаны к определенным типам данных.

В JavaScript есть семь основных типов данных:

 1.  **Строковый (string)** – любой текст, числа, символы, заключенные в одинарные или двойные кавычки - ‘Привет! ’; ’3,1415’; ’38 попугаев’

1. **Числовой (number)** – любые числа и результаты математических выражений. Числовые данные бывают двух типов:

* целочисленные, например - 123. Эти числа могут быть как положительными, так и отрицательными.
* вещественные числа, такие как 123,12. Их еще называют числа с плавающей точкой, они также могут быть положительными и отрицательными.

1. **Логический** или **булев (boolean)** – результаты выполнения любых условных операторов. Принимает два значения – **true** – истина, **false** – ложь
2. **Объект (object)** – сложный тип данных, представляет набор различных данных простых типов и функций, выполняющих с этими данными те или иные действия. Простейшее определение объекта представляют фигурные скобки: let user = {};

Объект может иметь различные свойства и методы:

let user = {name: "Иван", age:18};

    console.log(user.name);

5**.Символ (symbol)** – используется для создания уникальных идентификаторов объектов;

Кроме перечисленных типов данных, в JavaScript существуют еще два значения, которые тоже можно отнести к типам данных:

6. **undefined** -  значение, которое получает каждая новая переменная до присвоения ей какого – либо определенного значения;

7. **null** – означает отсутствие определенного значения.

В некоторых случаях возникает необходимость определить тип результата того или иного выражения. Для этого в языке JavaScript существует функция **typeof** (). Для любого выражения в скобках**,** результатом данной функции будет строка, содержащая название типа.

Чаще всего операторы и функции автоматически приводят переданные им значения к нужному типу. Например, **alert** автоматически преобразует любое значение к строке. Математические операторы преобразуют значения к числам.

Бывают случаи, когда нужно явно преобразовать значение в ожидаемый тип.

*Строковое преобразование* происходит, когда требуется представление чего – либо в виде строки. Например, **alert(value)** преобразует значение к строке. Также можно использовать функцию **String(value),** чтобы преобразовать значение к строке.

*Численное преобразование* можно осуществить через функцию **Number(value),** чтобы явно преобразовать **value** к числу:

let str = "123";

alert(typeof str); // string

let num = Number(str); // становится числом 123

alert(typeof num)

**Билет 4.** Виды диалоговых окон JavaScript. Функции их вызова

Существует три вида диалоговых окн:

1. Диалоговое окно предупреждения
2. Диалоговое окно запроса
3. Диалоговое окно подтверждения

Взаимодействие пользователя с браузером можно осуществлять через функции его интерфейса – alert, prompt и confirm.

**alert** – выводит сообщение и приостанавливает дальнейшее выполнение скриптов до тех пор, пока пользователь не нажмёт OK. (первый пример в учебном пособии). Сообщение выводится в модальном окне.

**prompt** – функция принимает два аргумента: result = prompt(title, [default]); Код отобразит модальное окно с текстом, полем для ввода текста и кнопками ОК/Отмена.

<script>

let name = prompt("Введите имя, пожалуйста", "Введите имя");

document.write("Привет, "+name+". Спасибо, что зашли на сайт!")

</script>

**confirm** – отображает модальное окно с текстом вопроса и двумя кнопками – ОК/Отмена.

<script>

Confirm(«text»);

</script>

Указанные методы имеют два ограничения:

1.Расположение окон определяет браузер (обычно центр);

2.Визуальное отображение окон зависит от браузера.

**Билет 5**. Условные конструкции языка JavaScript, их синтаксис.

В JavaScript имеются следующие условные конструкции (операторы):

1. Оператор **if** используется тогда, когда вы хотите, чтобы блок команд был выполнен если указанное условие истинно. Синтаксис:

if (условие) {

 блок команд;

}

Сначала проверяется **условие**, затем, если оно верно, выполняются **блок команд**, если оно не верно, то блок команд пропускается, и продолжается выполнение сценария.

2. Конструкция **if…else** используется, если вы хотите, чтобы один блок команд выполнялся если указанное условие истинно и другой блок команд, когда условие ложно. Синтаксис:

if (условие) {

операторы 1;

}

else {

операторы 2;

}

**Билет 6. Операторы цикла JavaScript, их синтаксис. Операторы break и continue.**

**Цикл** – это блок команд, который может повторно выполняться пока определенное условие не будет выполнено.

Цикл **for** использует блок команд пока заданное условие является истинным. Синтаксис:

for (выражение инициализации; выражение условия; выражение цикла)

{

операторы

}

Цикл **while** сходен с циклом **for**, но он не производит инициализацию и инкремент счетчика в своем объявлении. Синтаксис:

let i=1;

while (i<=5) {

   document.write (i+'<br />');

      i++;

}

Цикл **do…while** часто называют циклом с постусловием, потому что в отличие от предыдущих циклов он вначале исполняет блок команд и только потом проверяет заданное условие.

let i = 0;

do {

alert( i );

i++;

} while (i < 3);

Иногда в ходе выполнения цикла возникает необходимость прервать весь цикл или одну его итерацию. Для этой цели служат операторы **break** и **continue**. Оператор **break** полностью прекращает выполнение цикла и предает управление операторам, следующим за циклом. Оператор **continue** прерывает текущее выполнение цикла и переходит к выполнению следующего шага цикла.

**Билет 7**. Массивы в JavaScript. Основные виды, способ организации. Свойства и методы

**Массив** – это переменная, в которой в упорядоченном виде можно хранить целый набор значений. Массив создается с помощью квадратных скобок [ ].

В одном массиве можно хранить различные типы данных, массивы JavaScript не типизированы.

Элементами массива могут быть и другие массивы, т.е. получаем **массив массивов** или **многомерный массив**.

Создадим массив студентов **students**, который содержит два подмассива: студенты юноши и студенты девушки.

let students = {

    'boys': ['Коля', 'Вася', 'Петя'],

    'girls': ['Даша', 'Маша', 'Лена'],

}

alert(students['boys'][0]);

Чтобы вывести какой – либо элемент из многомерного массива, следует писать не одну пару [ ] , а две: **students[**'**boys**'**][0]** – выведем Колю.

**Метод** **push** – добавляет неограниченное количество элементов в конец массива. При этом исходный массив изменяется, а результатом возвращается новая длина массива.

**Пример**: добавим два новых элемента в конец массива:

let arr = ["ПН ", "ВТ ", "СР ", "ЧТ ", "ПТ "];

arr.push("СБ ", "ВС ");

document.write(arr); // ПН ,ВТ ,СР ,ЧТ ,ПТ ,СБ ,ВС

**Метод unshift** – добавляет неограниченное количество новых элементов в начало массива. При этом исходный массив изменяется, а результатом возвращается новая длина массива.

**Пример**: добавим два новых элемента в начало массива:

let arr = ["ПН ", "ВТ ", "СР ", "ЧТ ", "ПТ "];

arr.unshift("СБ ", "ВС ");

document.write(arr); // СБ ,ВС ,ПН ,ВТ ,СР ,ЧТ ,ПТ

**Метод** **shift** – удаляет первый элемент из массива.

**Метод pop** – удаляет последний элемент из массива.

**Метод splice** – позволяет **удалять, добавлять и производить замену** элементов в массиве.

***Удаление элементов***: в   методе указывается два числовых параметра (1 – с какого элемента начать удаление, 2 – количество удаляемых элементов.

***Добавление элементов***: чтобы добавить элементы в массив с помощью метода splice также необходимо использовать параметры (1 – число перед каким элементом массива следует начинать добавление, 2 – число 0. После этого указываем элементы, которые необходимо вставить в указанную позицию).

***Замена элементов*** осуществляется путем удаления элемента, а затем добавления новых.

**Метод concat** – позволяет объединить массивы

**Свойства** – это пара ***«ключ: значение»***, где ***ключ*** – это строка (имя свойства), а ***значение*** может быть, чем угодно.

**Создание объекта:**

let user = {

name: "John",

age: 30

};

У каждого свойства есть ключ (имя или идентификатор). После него ставится двоеточие, а затем указывается значение свойства. Если в объекте несколько свойств, то они перечисляются через запятую.

Для обращения к свойствам используют запись «через точку»:

document.write(user.name);

document.write(user.age);

Билет 8. События в JavaScript. Пользовательские функции. Синтаксис создания функции

**Событие** – это сигнал от браузера о том, что что-то произошло.

Существует три основных типа событий:

1.События мыши

2.События на элементах управления

3.Клавиатурные события

**События мыши:**

Clik-Щелчок левой кнопкой мыши по элементу (на устройствах с сенсорными экранами оно происходит при касании)-onclick

<script>

function myFc() {

alert("Вы щелкнули по тексту");

}

</script>

<div onClick="myFc();">Щелкни меня</div>

**События на элементах управления:**

Submit-Нажатие кнопки Отправить в форме-onSubmit

<form onsubmit="alert('submit!');return false">

нажмите на кнопку "Отправить": <input type="submit" value="Отправить">

</form>

**Клавиатурные события:**

|  |  |  |
| --- | --- | --- |
| keydown/ keyup | Нажатие/ отпускание клавиши клавиатуры | onKeyDown/ onKeyUp |

**Билет 9. Назначение языка PHP. Возможности PHP. Основы синтаксиса**

**PHP** - это серверный язык создания сценариев. Конструкции PHP, вставленные в HTML-текст, выполняются сервером при каждом посещении страницы. Результат их обработки вместе с обычным HTML - текстом передается [браузеру](http://click02.begun.ru/click.jsp?url=P0yJx3J5eHmBEHO86rMyxK44ZLs3R*zH*tbaEsuCNfO6n3GPmh-WC09etu8ZdzofC8HDEDbwNInV0PSfyIYo8d2nlnmfiK*apExJ4dpK4nwQQNfrmVGhFVWDr9DEwk5spR557MFKKncNimiYEXSbkzewPTByCaMeqz1si6oSoNs0iMYJjSVHuwLbkOeATwVjs1AfqN39id0r-g7xoBvd2zm20YZ8CnRn7P8LupyjyzF0bC-wpcsxUbRHVQ8Da9FcJd39n1TpvF2T15QwbLtx4WuvFORTzIFWboJhDPTvfi-ngeCfUESDTLfpY8HY*03Qh*gWXqo0lTThVny1AdbKW4EfLbZ0s0IqRDGhTRJwTHMti5i-iNVYDE0KSqXVzP3L1cYjc5SiHGt6tt4X8S1887KAzAguWXd8HeaVZ6XUWvnjX2TRD8zw0cRHwHHjfyF-dOUIFHqzDYcEn7gSxUYXB68e12w&eurl%5B%5D=P0yJx11cXVykc9-MBLsWp59AKRWJTlgGscZWMBsMRgKYIxoA).

PHP-программы состоят из простого текста, поэтому набирать их можно в любом текстовом редакторе. Расширение файлов PHP - программ по умолчанию - \***.php** - на основании этого расширения сервер распознает файл как PHP-программу и запускает интерпретатор.

PHP-программа должна быть отделена от обычного HTML-текста следующей конструкцией:

<?php

Код

?>

**Возможности PHP**

1. PHP используется в связке с БД, которые являются хранилищами информации. Язык PHP позволяет работать с различными типами БД, такими как MySQL, Oracle, InterBase и другие;
2. Широко используется возможность работы с электронной почтой. Достаточно несложно создать PHP – приложения для рассылки и приема писем;
3. С помощью PHP можно формировать изображения, файлы типа PDF, а также flash – анимации;
4. Работа с файлами также является частью языка. PHP предоставляет полный спектр функций с помощью которых можно удалять, копировать, создавать файлы, записывать и извлекать из них нужную информацию;
5. PHP имеет возможность работать с технологиями в области электронной коммерции.

**Билет 10.  Переменные, константы и типы данных в PHP**

**Переменная** – средство языка для хранения данных.

Имя любой переменной в PHP начинается со знака **$.** Имена переменных чувствительны к регистру символов. Имя переменной может содержать только латинские буквы, числа, (\_). Имя переменной не может начинаться с числа.

**Константы**

Константа содержит значение, которое нельзя изменить на протяжении всего сценария. Для ее объявления следует использовать функцию **define( )**. Например:

define(PI, 3.14, true)

здесь объявлена константа с именем PI, которая имеет значение 3.14. Третий параметр (необязательный) указывает, будет ли константа чувствительна к регистру. В нашем случае регистр символов учитывается.

Принято, имена переменных обозначать строчными буквами, имена констант – прописными.

PHP поддерживает четыре скалярных типа данных:

* **integer**- целое число;
* **double** – число с плавающей точкой;
* **boolean**- логический тип данных (принимает значения true или false);
* **string –** строковый тип данных;

и два смешанных типа данных:

* **array –** массив;
* **object -** объект.

**Билет 11.** Управляющие конструкции в PHP, их синтаксис.

**Условный оператор**

Условный оператор **if – else** имеет следующий синтаксис:

**if (условие) действие1;**

**if (условие) действие1;**

**else действие2;**

1. **if (условие) действие1;**

**elseif (условие) действие2;**

**else действие3;**

### Функция empty

Функция **empty** осуществляет проверку переменной на **пустоту**. Переменная будет пуста, если она равна нулю, ‘’ (пустой строке), **false** или **null** (не определена заранее). Пример использования:

<?php $a **=** '';

//Если $a пустое, то...

if (empty($a)) echo 'Верно!'; else echo 'Неверно!'; //выведет 'Верно!'

?>

Чаще возникает обратная задача – проверка на то, что переменная является **НЕ пустой**. Отрицание НЕ делается с помощью оператора ! Пример применения:

<?php $a **=** null;

//Если $a Непустое, то...

if (**!**empty($a)) echo 'Верно!'; else echo 'Неверно!';

//выведет 'Неверно!', так как $a пустое

?>

### **Оператор выбора**

Оператор выбора **switch** оценивает одно выражение и в зависимости от его значения выполняет один из нескольких блоков программы. Синтаксис:

**switch (выражение)**

**{**

**case значение1:**

**действие1;**

**break;**

**case значение2:**

**действие2;**

**break;**

**…**

**default:**

**действиеN;**

**}**

Пример использования оператора switch: выведем список статей за конкретный год.

<?php

$year = 2014;

switch($year)

{

 case 2014:

    echo 'Год 2014<br>';

    echo 'Список статей<br>';

 break;

  case 2015:

    echo 'Год 2015<br>';

    echo 'Список статей<br>';

 break;

  default:

 echo 'Неправильно указан год <br>';

}

?>

**Цикл с предусловием**

**while (условие)**

**{**

**тело цикла;**

**}**

 При входе в цикл проверяется условие и, если его значение верно, выполняется тело цикла. Так тело цикла будет выполняться до тех пор, пока условие не станет ложным. Таким образом, тело цикла может быть не выполнено вообще, если изначально условие ложно. Пример использования цикла с предусловием:

<?php

$a = 10;

while ($a >1)

  {

   echo --$a.' ';

   }

?>

Сначала $a присваивается значение 10. Затем выполняем цикл с условием, пока $a>1. В теле цикла мы уменьшаем значение переменной на 1 и выводим на экран с последующим символом пробела: 9 8 7 6 5 4 3 2 1.

**Цикл с постусловием**

**do {**

**тело цикла;**

**}**

**while (условие);**

Этот тип цикла абсолютно такой же, как и предыдущий, но с одним явным отличием – сначала выполняется тело цикла, а только затем проверяется условие выхода. В этом случае при любом условии, даже заведомо ложном, тело цикла будет выполнено хотя бы один раз. Пример использования цикла с постусловием:

<?php

$a = 10;

do  {

   echo $a;

  }

while ($a !=10);

?>

В результате выполнения браузер выведет число 10.

**Билет 12. Организация передачи данных пользователя через форму в PHP. Методы передачи данных.**

**Форма** *—* это особый элемент страницы, выполняющий кодирование данных и пересылку их веб - серверу. Сами же элементы управления только принимают данные от посетителя, но не кодируют и не передают их.

Каждый элемент управления, находящийся в форме, должен иметь уникальное имя

**ДОПИШИТЕ ПЖПЖ**

**Билет 13.** Подключение внешних файлов средствами PHP.

**Чтение файла**

Функция **file\_get\_contents** позволяет выполнить чтение файла. Параметром функция принимает имя файла, а результатом возвращает текст этого файла.

Пусть у нас есть файл index.php, к которому будем обращаться через браузер. Пусть в этой же папке находится файл text.txt.

Запишем в файл text.txt любую фразу, например, *«Привет из файла»* и выведем этот текст на экран:

**Запись файла**

Для записи данных в файл предназначена функция **file\_put\_contents**, которая первым параметром принимает путь к файлу, вторым – текст, который хотим записать.

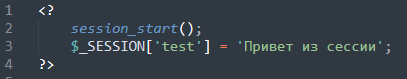
**Билет 14. Сессии и куки**

**Сессия** - это механизм PHP, который позволяет хранить данные для конкретного пользователя между запусками скрипта.

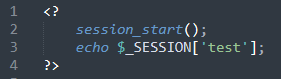
С помощью сессии можно реализовать авторизацию пользователей, корзину интернет-магазина и другое.

**Инициализация сессии** начинается с функции **session\_start.** После инициализации в сессию можно что-то **записать** или что-то **прочитать** из нее. Делается это с помощью массива $\_SESSION.

В файл test1.php организуем сессию и запишем в нее текст «Привет из сессии»:



В файле test2.php информацию из сессии можно **извлекаем**:



**Особенности работы с сессиями:**

1. Нельзя делать никакого вывода в браузер до окончания работы с сессиями;
2. Кодировка документа должна обязательно быть **utf\_8 без BOM**.

**Cookie** (куки) – это способ долговременного хранения данных в браузере пользователя.

Написать что-то в куки можно с помощью функции **setcookie**, которая первым параметром принимает имя этой куки, а вторым – значение:



**Время жизни куки в PHP**

Время жизни куки задается с помощью третьего параметра, который принимает момент времени в формате **timestamp.** Как только этот момент наступил – кука автоматически будет удалена браузером.

 Третий параметр принято записывать так: настоящее время + N секунд. Формат **timestamp** можно получить с помощью функции **time**.

**Билет 15.** Основы работы с MySQL через PHP

**Основные функции для работы с БД:**

Работа с БД из PHP осуществляется с помощью следующих функций:

**mysqli\_connect** – соединение с сервером и БД;

**mysqli\_close** – закрытие соединения с сервером;

**mysqli\_query** – отправка запроса к БД;

**mysqli\_error** – вывод ошибок.

**Соединение с базой данных из PHP (mysqli\_connect)**

Функция принимает параметры:

* **имя хоста** (сервера);
* **имя пользователя**, под которым мы работаем с БД;
* **пароль** для этого пользователя;
* **имя БД**.

Для соединения с БД через PHP – файл необходимо прописать следующий код:

**mysqli\_connect(“сервер”, “пользователь”, “пароль”, “имя БД” );**

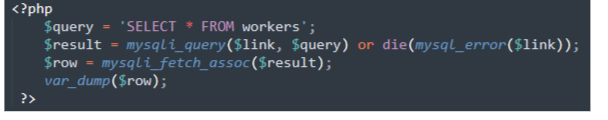
**Посылка запросов к БД:**

После соединения с базой к ней можно отправлять запросы. Делается это с помощью функции **mysqli\_query**. Первым параметром указывается переменная, в которую записано соединение к БД, в нашем случае это $**link**, вторым – строка с SQL запросом.

**Вывод данных из таблицы:**

После того, как сделан запрос к БД, в переменной **$result** будет лежать результат этого действия. Но результат представлен не в той форме, которая нам нужна в PHP, а в той, в которой его прислала база данных.

Для того, чтобы получить результат в привычной нам форме, необходимо воспользоваться функцией **mysqli\_fetch\_assoc**, извлекающей из результата одну строку.

****