# Билет 1

# 1. LESS, SASS, SCSS. Отличия от CSS, особенности синтаксиса, особенности работы на старых браузерах

## LESS

**LESS** - это динамический язык стилей, который является надстройкой над CSS (Поэтому любой CSS код будет валидный LESS).

Приемущества LESS:

1. Переменные (и области видимости переменных).
2. Операции (в том числе и для управления цветом, т.е можно смешивать цвета: #941f1f + #222222).
3. И другие функции для работы с цветом (осветление, затемнение и т.п.)
4. Вложенность (можно вложить одно правило в другое, article.post p {} <=> article.post { p{ }}).
5. Объединение аргуметов.

LESS-файл конвертируется в CSS при помощи js (для этого необходимо скачать less.js с сайта LESS).

## Sass

**Sass** - это метаязык на основе CSS, предназначенный для увеличения уровня абстракции CSS кода и упрощения файлов каскадных таблиц стилей.

Приемущества Sass:

1. Вложенные правила.
2. Переменные.
3. Возможность создавать миксины, позволяющие создавать многоразовые CSS-правила - группы деклараций, для многократного использования. (LESS в это не может)
4. Расширения. Одиночный селектор может быть расширен больше, чес одним селектором с помошью @extend.
5. Есть логика. (if/then/for). (Этого в LESS тоже нет)

6. Не может компилироватся на сервере в CSS (LESS использует js).

Браузер не распознает файлы Sass, так что сначала их нужно скомпилировать в обычный CSS.

## SCSS

**SCSS** — "диалект" языка SASS. Отличие SCSS от SASS заключается в том, что SCSS больше похож на обычный CSS код.

1. @import - @import "template" подключит template.scss.
2. Вложенность.
3. $переменные.
4. Математика чисел и цветов.
5. Строки (умеет складвать строки, поддерживает конструкцию #{$var})

# 2. Java SE, Java EE, Java ME. Архитектура приложений Java EE, понятие о контейнере, компоненте.

- Java SE — основные инструменты и библиотеки языка
- Java EE — платформа для enterprise серверных приложений (JSP, JSF, сервлеты и т.д.)
- Java ME — урезанная платформа для мобильных устройств (в основном использовалась до появления iOS/Android)

## Архитектура приложений Java EE, понятие о контейнере, компоненте.

Сервлеты — серверные сценарии, жизненным циклом которых управляет веб-контейнер. Существует несколько веб-контейнеров (Apache Tomcat, GlassFish, Jetty, др.); они все реализуют определенный стандарт Java EE (6/7/8).

При запуске веб-контейнер читает манифест (web.xml) приложения и создает для него ServletContext. Для каждого класса сервлета и фильтра из манифеста создается объект. Контекст и объекты фильтров/сервлетов будут жить на протяжении работы приложения, т.е. одна и та же инстанция будет использоваться для всех запросов и сессий.

При поступлении HTTP запроса веб-контейнер создает новые объекты HttpServletRequest и HttpServletResponse и передает их фильтрам, потом сервлету.

Компоненты - классы, написанные по определённым правилам, которые дают возможность создания инструментов.

# Билет 2

## 1. Структура HTTP-запроса

HTTP запрос состоит из трех основных частей, которые идут в нем именно в определенном порядке:

1. строка запроса (Request Line) - указывает метод передачи, URL-адрес, к которому нужно обратиться и версию протокола HTTP.
2. заголовки (Message Headers) - описывают тело сообщений, передают различные параметры и др. сведения и информацию.
3. пустая строка (разделитель)
4. тело сообщения (Entity Body) – необязательный параметр - данные, которые передаются в запросе.

### Методы запросов:

- `GET` запрашивает представление ресурса, может только извлекать данные.
- `HEAD` как GET, только без тела ответа.
- `POST` используется для отправк сущностей определенному ресурсу. Может изменять данные.
- `PUT` создает новый ресурс или заменят предсавление целевого ресурса (в отличае от POST для идентичных наборов двнных будет иметь одинаковый результат).
- `DELETE` удаляет ресурм.
- `CONNECT` устанавливает "туннель" к серверу, определенному по ресурсу.
- `OPTIONS` для описания параметров соединения с ресурсом
- `TRACE` вызов возвращаемого текстового сообщения
- `PATCH` частичное изменение ресурса

## 2. Типы данных PHP

PHP - яп с динамической типизацией. Преобразования меджду скалярными типами происходят неявно.

Скалярные типы: `integer`, `float`, `double`, `boolean`, `string`.

Нескалярные: `array`, `object`, `resource`, `null`

Псевдотипы: `mixed` (любой тип), `number`, `callback` (string или анонимная фунуция), `void`.

# Билет 3

## Методы HTTP

Метод HTTP (англ. HTTP Method) — последовательность из любых символов, кроме управляющих и разделителей, указывающая на основную операцию над ресурсом. Обычно метод представляет собой короткое английское слово, записанное заглавными буквами. Обратите внимание, что название метода чувствительно к регистру.

- `OPTIONS`: Используется для определения возможностей веб-сервера или параметров соединения для конкретного ресурса. В ответ серверу следует включить заголовок `Allow` со списком поддерживаемых методов. Также в заголовке ответа может включаться информация о поддерживаемых расширениях.
- `GET`: Используется для запроса содержимого указанного ресурса. С помощью метода `GET` можно также начать какой-либо процесс. В этом случае в тело ответного сообщения следует включить информацию о ходе выполнения процесса.
- `HEAD`: Аналогичен методу `GET`, за исключением того, что в ответе сервера отсутствует тело. Запрос HEAD обычно применяется для извлечения метаданных, проверки наличия ресурса (валидация URL)
- `POST`: Передача пользовательских данных заданному ресурсу
- `PUT`: Загрузка содержимого запроса на указанный URI (если такого не было - код возврата `201`: Created)
- `PATCH`: Аналогичен `PUT`, но применяется к фрагменту ресурса
- `DELETE`: Удаляет определенный ресурс
- `TRACE`: Возвращает полученный запрос, таким образом есть возможность увидеть информацию, добавленную промежуточными серверами
- `CONNECT`: Преобразует соединение запроса в прозрачный TCP/IP-туннель

## 2. Жизненный цикл сервлета

Жизненный цикл сервлета состоит из следующих шагов:

- В случае отсутствия сервлета в контейнере.

1. Класс сервлета загружается контейнером.
2. Контейнер создает экземпляр класса сервлета.
3. Контейнер вызывает метод `init()`.

- Обслуживание клиентского запроса. Каждый запрос обрабатывается в своем отдельном потоке. Контейнер вызывает метод service() для каждого запроса. Этот метод определяет тип пришедшего запроса и распределяет его в соответствующий этому типу метод для обработки запроса. Разработчик сервлета должен предоставить реализацию для этих методов. Если поступил запрос, метод для которого не реализован, вызывается метод родительского класса и обычно завершается возвращением ошибки инициатору запроса.
- В случае если контейнеру необходимо удалить сервлет, он вызывает метод `destroy()`, который снимает сервлет из эксплуатации. Подобно методу init(), этот метод тоже вызывается единожды за весь цикл сервлета.ser

# Билет 4

## 1. Коды состояния HTTP

Коды состояния -- часть ответа сервера по протоколу HTTP. Их особенности: Состоят из 3-х цифр. Первая цифра — класс состояния: «1» — Informational — информационный; «2» — Success — успешно; «3» — Redirection — перенаправление; «4» — Client error — ошибка клиента; «5» — Server error — ошибка сервера.

## 2. Суперглобальные массивы (Superglobal Arrays) в PHP

предопределенные (не надо явно указывать global
$var) массивы (и переменные), имеющие глобальную область видимости$ :
GLOBALS (которые доступны в любом месте скрипта).

`$_SERVER` : заголовки, пути и местоположения скриптов.

`$_GET` : переменные GET-запроса

`$_POST` : переменные POST-запроса

`$_FILES` : переменные файлов, загруженных по HTTP

`$_COOKIE` : переданные скрипту через HTTP Cookies.

`$_SESSION` : переменные сессии

`$_REQUEST` : по умолчанию содержит данные переменных `$_GET`, `$_POST` и `$_COOKIE`

`$_ENV` : переменные окружения

# Билет 5

## 1. Структура HTML-документа

Документ состоит из элементов, начало и конец которых обозначаются тегами. Некоторые теги не могут содержать текст (например, `<br>` — перенос строки, `<img>` — картинка, `<input>` — элемент ввода в форме). Их не нужно закрывать:

В HTML5 введены семантические теги `<header>`, `<footer>`, `<section>`, которые аналогичны `<div>`, но указывают на логическую структуру.

# Билет 7

## 1. DOM и BOM

**DOM** - программный интерфейс для HTML и XML документов, описывающий структурированное представление документа и определяющий, как это структура может быть доступна из программ, которые могут изменять ее содержимое.

Согласно DOM документ может быть представлен в виде объектов, с которыми можно проихводить манипуляци:

1. генерация и добавление узлов
2. получение узлов
3. изменение узлов
4. изменение связей между ними
5. удаление узлов

**BOM** - объектная модель браузера. Основное предназначение — управление окнами браузера и обеспечение их взаимодействия. BOM специфична для каждого браузера. Может создавать системные диалоги, управлять информацией о параметрах монитора и браузера и т.д.

Обращение к узлам происходит с помощью элементов document (DOM) или window (BOM).

## 2. Управление сессией. HttpSession

Сеанс (сессия) – соединение между клиентом и сервером, устанавливаемое на определенное время, за которое клиент может отправить на сервер сколько угодно

запросов. Сеанс устанавливается непосредственно между клиентом и Web-сервером. Каждый клиент устанавливает с сервером свой собственный сеанс.

Чтобы открыть новый сеанс, используется метод `getSession()` интерфейса `HttpServletRequest`. Метод извлекает из переданного в сервлет запроса объект сессии класса `HttpSession`, соответствующий данному пользователю.

Чтобы сохранить значения переменной в текущем сеансе, используется метод `setAttribute()`, прочесть – `getAttribute()`, удалить – `removeAttribute()`. Список имен всех переменных, сохраненных в текущем сеансе, можно получить, используя метод `Enumeration getAttributeNames()`.

Есть 3 способа отслеживания сессии: cookies, переопределяемый URL (используется response.encodeURL() для каждой ссылки, который вставляет идентификатор сессии в каждый URL.), скрытые поля форм.

# Билет 8

# 1. CSS : назначение, правила, приоритеты

**CSS** - формальный язык описания внешнего вида документа с помощью языка разметки.

**Селекторы** - имя тега, к которому применяется правило.

## Основные виды селекторов:

- `*` - любые элементы
- `div` элементы с тегом div
- `#id` - элемент по id
- `.class` - элементы с классом class
- `[name="value"]` - селекторы по атрибуту
- `:visited` - псевдоклассы
- `div p` - элементы p, являющиеся потомками div
- `div > p` – только непосредственные потомки
- `div ~ p` – правые соседи: все p на том же уровне вложенности, которые идут после div
- `div + p` – первый правый сосед: p на том же уровне вложенности, который идёт сразу после div

## Приоритеты правил.

1. Самый высокий приоритет имеет атрибут style.
2. Второе по приоритету - присутствие ID в селекторе.
3. Все атрибуты (включая class и псевдоклассы)
4. Самый низкий - селекторы с именами элементов и псевдоэлементами.

`!important` позволяет повысить приоритет стиля.

# 2. MVC : назначение, элементы, примеры реализации

**Model-View-Controller** — схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

- Модель (Model) предоставляет данные и методы работы с ними: запросы в базу данных, проверка на корректность (не зависит от представления (не знает как данные визуализировать) и контроллера (не имеет точек взаимодействия с пользователем).
- Представление (View) отвечает за получение необходимых данных из модели и отправляет их пользователю (не обрабатывает введённые данные пользователя).
- Контроллер (Controller) обеспечивает «связи» между пользователем и системой. Контролирует и направляет данные от пользователя к системе и наоборот.

```
// Контролеер
class CalcController : IController {

  public void OnClick( int number ) {
        view.Total = model.SetInput(number).ToString();
    }

    public void OnAdd() {
        model.ChangeToAddState();
    }
}
```

```
// Представление
public class frmCalcView : Form, ICalcView{
  public string Total{
    get{
        return textBox1.Text;
    }
```

```
    set{
        textBox1.Text = value;
    }
  }
}
```

```
// Модель
class CalculatorModel : ICalcModel{
  public int SetInput ( int number ) {
      if (state == States.NoOperation) {
          currentValue = number;
      }
      else if (state == States.Add){
          currentValue = Add(currentValue , number );
      }
      return currentValue;
  }
}
```

# Билет 9

## 1. AJAX и DHTML - описание, сходства и различия

**Asynchronous Javascript and XML** - подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером и динамическом обновлением контента.

Реализация AJAX с подощью JQuery:

```
$.ajax({
  type: "POST",
  url: "some.php",      // обращение к some.php
  data: {name: 'John', location: 'Boston'},   //с какими-то параметрами
  success: function(msg){
    alert( "Data Saved: " + msg );  // получный результат выводится в
alert
  }
});
```

**DHTML** - технология создания интерактивных динамических веб-страниц с использованием стандартных средств: html, css, DOM и js. Построен на DOM.

```
<html>
<head>
    ...
</head>
<body>
<H2>Всплывание события</H2>
<b>Для получения информации можно нажать на изображение или текст</b>
<br>
<a href="kuda.htm"><img src="be.gif" align="left">Кто это?</a>
</body>
</html>
```

AJAX позволяет не перезагружать открытую пользователем страницу целиком при внесении в неё каких-либо небольших изменений сервером, а ограничиться загрузкой только небольшого фрагмента.

AJAX-страницы для придания им большей интерактивности реализуются с использованием DHTML, однако DHTML-страницы совершенно не обязательно используют AJAX.

DHTM это различные выпадающие меню на сайтах, разворачивающиеся по клику пользователя панели и древовидные списки, и т. п. Технически Dynamic HTML реализуется с помощью JavaScript, CSS и DOM.

## 2. Какие проблемы возникают при параллельной обработке запросов в JSP, как этого можно избежать?

Согласно спецификации сервлетов, web контейнер по умолчанию позволяет одному и тому же экземпляру сервлета параллельно обрабатывать запросы сразу от нескольких клиентов. При этом каждому из запросов выделяется отдельный трэд.

За это отвечает директива `isThreadSafe`, дефолтное значение которого - `true`. Для того, чтобы запретить параллельную обработку запросов необходима директива `isThreadSafe` со значением `false`.

# Билет 10

# 1. Rest и RPC

**REST** - архитектурный стиль взаимодействия компонентов приложения в сети.

Требования к архитекруте REST:

1. Модель клиент-сервер
2. Отсутствие состояния (в период между запросами клиента никакая информация о состоянии клиента на сервере не хранится)
3. Кэширование (клиенты могут выполнять кэширование ответов с сервера, ответы должны иметь обозначения как кэшируемые или нет для избежания получения неверных последующих ответов)
4. Единообразие интерфейса
5. Слои (сервера, какие-то промежуточные сервера и узлы)

**RPC** - удалённый вызов процедур. Что-то вроде совокупности клиент-серверного приложения с сериализацей объектов.

# 2. RequestDispatcher

Штучка, которая работает на уровне контейнера сервлетов. Контейнер создает объект RequestDispatcher - обертка сервлета, с помощью которого можно потом получать запросы от клиента и перенаправлять их.

```
RequestDispatcher rd = request.getRequestDispatcher("servlet2");
// в каком-то контроллере сервлетов создаем объект RequestDispatcher
другого сервлета
// вообще, есть 2 способа получения request - через ServletRequest или
через контекст приложения ServletContext

rd.forward(request, response);
// для обработки запроса есть еще метод include
```

# Билет 16:

# 1. Элементы JSP

## Комментарии

- комментарии исходного кода JSP: `<%-- этот комментарий удаляется на этапе компиляции jsp-страницы --%>`

- комментарии HTML-разметки: `<!-- рассматривается как статический текст и помещается в HTML-документ -->`
- java-комментарии

## Директивы

Для передачи аргументов и параметров контейнеру. `<%@ атрибут="значение" %>`

## Объявления

Объявления используются для **определения** используемых в программе конструкций Java (задавать переменные, методы, внутренние классы).

```
<%! private int accessCount = 0; %>
Количество обращений к странице с момента загрузки сервера: <%=
++accessCount %>
```

## Скриптлеты

Скриплеты JSP дают возможность вставить любой код в метод сервлета, который будет создан при обработке страницы, позволяя использовать большинство конструкций Java.

```
<%
String queryData = request.getQueryString();
out.println("Дополнительные данные запроса: " + queryData);
%>
```

## Выражения

Выражения JSP применяются для того, чтобы вставить значения Java непосредственно в вывод. Выражения Java вычисляются, конвертируются в строку и вставляются в страницу.

```
Текущее время: <%= new java.util.Date() %>
Имя вашего хоста: <%= request.getRemoteHost() %>
```

# 2. CGI - обработка запроса, преимущества и недостатки

CGI (Common Gateway Interface — общий интерфейс шлюза) — стандарт интерфейса, используемого для связи внешней программы с веб-сервером. Программу, которая работает по такому интерфейсу совместно с веб-сервером, принято называть шлюзом (оно же скрипт или CGI-программа). По сути позволяет использовать консоль ввода и вывода для взаимодействия с клиентом.

## CGI - сценарии

- CGI — механизм вызова пользователем программ на стороне сервера.
- Данные отправляются программе посредством HTTP-запроса, формируемого веб-браузером.
- То, какая именно программа будет вызвана, обычно определяется URL запроса.
- Каждый запрос обрабатывается отдельным процессом CGI-программы.
- Взаимодействие программы с веб-сервером осуществляется через stdin и stdout.

# Билет 11

# 1. Особенности языка JS: динамическая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса

## Динамическая типизация

Позволяет не указывать тип, но при этом язык не определяет его сам. Типы переменных неизвестны до того момента, когда у них появляется конкретное значения при запуске.

## Автоматическое управление памятью

Память автоматически выделяется при объявлении переменных, вызове некоторых функций/методов. Сборщик мусора освобождает память, когда больше не существует ссылок, позволяющих добраться до объекта от корня

## Прототипное программирование

Стиль объектно-ориентированного программирования, при котором отсутствует понятие класса, а наследование производится путём клонирования существующего экземпляра объекта — прототипа.

## Объекты первого класса

Объекты первого класса - это элементы, которые могут быть переданы как параметр, возвращены из функции или присвоены переменной. Определение функции (не вызов!) — это выражение, а значит оно возвращает значение, а именно — функцию, что доказывает, что функции в JS — объекты первого класса

# 2. Long Polling и WebSockets -- что такое, для чего нужно, преимущества и недостатки друг относительно друга

## Long Polling

Long Polling — это технология, которая позволяет получать данные о новых событиях с помощью «длинных запросов». Сервер получает запрос, но отправляет ответ на него не сразу, а лишь тогда, когда произойдет какое-либо событие (например, придёт новое сообщение), либо истечет заданное время ожидания.

1. Клиент запрашивает страницу у сервера, используя обычный http
2. Запрошенная страница выполняет JavaScript, который запрашивает файл от сервера.
3. Сервер НЕ реагирует на запрошенную информацию и ждет, пока не появится новой информации
4. Когда появляется новая информация, сервер отсылает ее клиенту
5. Клиент получает новую информацию и сразу отсылает другой запрос серверу, запуская процесс ожидания на нем снова.

## WebSokets

WebSocket — протокол полнодуплексной связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и вебсервером в режиме реального времени.

1. Клиент запрашивает страницу у сервера, используя обычный http
2. Открывается соединение с сервером
3. Сервер и клиент могут посылать друг другу сообщения

## Преимущества WebSocket:

- Поддерживается всеми современными браузерами (даже IE).
- Не требуется рвать соединение при доставки сообщения.

- (Следствие второго) можно отправлять больше сообщений пользователю в секунду
- Можно самостоятельно отправлять серверу сообщения по WebSockets, то есть, общение двустороннее.

# Билет 17

## 1. FastCGI. Плюсы, минусы, отличия от CGI

CGI — устаревшая технология, позволяющая взаимодействовать веб-серверу с сервером приложений. Для каждого запроса запускается процесс с интерпретатором PHP, после возвращения ответа он завершается. Поскольку это очень неэффективно, был создан FastCGI, в котором процесс интерпретатора не завершается, а используется для последующих запросов. В этом есть и плюсы и минусы. К плюсам можно отнести перфоманс, к минусам то, что если что-то сломается, то в CGI упадет один процесс и (наверное) этого никто не заметит, в FastCGI упадет все. Кроме того, эта штука работает не через stdin/stdout, а что-то другое.

## 2. Суперглобальные массивы в PHP (SuperGlobal massive)

Есть предопределенные (не надо явно указывать global $var) массивы (и переменные), которые доступны в любом месте скрипта.

- `$GLOBALS` : ссылки на все переменные глобальной области видимости
- `$_SERVER` : заголовки, пути и местоположения скриптов.
- `$_GET` : переменные GET-запроса
- `$_POST` : переменные POST-запроса
- `$_FILES` : переменные файлов, загруженных по HTTP
- `$_COOKIE` : переданные скрипту через HTTP Cookies.
- `$_SESSION` : переменные сессии
- `$_REQUEST` : по умолчанию содержит данные переменных $_GET, $_POST и $_COOKIE
- `$_ENV` : переменные окружения

# Билет 18

## 1. ООП в PHP

```
class ClassName {
  public $publicName;
  private $privateName;
  protected $protectedName;

  const CONST_VAL = 'val';

  public function getPrivateName() {
    return $this→$privateName; // $this -- ссылка на сам объект,
$parent - на родительский.
  }
};

echo ClassName::CONST_VAL; // для обращения к константам

$classname = new ClassName();

$className→publicName; // доступ к переменной
```

PHP поддерживает все три основных механизма ООП — инкапсуляцию, полиморфизм подтипов и наследование (с помощью extend). Поддерживаются интерфейсы (с помощью implements). Есть абстрактные и final методы и классы. Множественное наследование не поддерживается, но класс может реализовывть несколько интерфейсов или с помощью механизма особенностей (trait), который имеет средства для разрешения конфликтов.

Методы: `__construct()` -- конструктор `__destruct()` -- для деинициализации объекта `__get()`, `__set()` `__sleep()`, `__wakeup()` `__clone()`

# 2. Предопределенные переменные JSP

- `request` : `HttpServletRequest` -- позволяет обращаться к параметрам запроса (через метод getParameter), типу запроса (GET, POST, HEAD) и входящим HTTP заголовкам (cookies, Referer)
- `response` : `HttpServletResponse`
- `out` : `PrintWriter` -- для отправки вывода клиенту
- `session` : `HttpSession`
- `application` : `ServletContext`
- `config` : `ServletConfig` -- для текущей страницы
- `page` : синоним `this`
- `pageContext`

# Билет ??

## 1. Особенности javascript

JavaScript является объектно-ориентированным языком.

Основные архитектурные черты:

- динамическая типизация; (автоматическое приведение типов)
- слабая типизация;
- автоматическое управление памятью; (автоматическая сборка мусора)
- прототипное программирование; (отсутствует понятие класса, а наследование производится путём клонирования существующего экземпляра объекта — прототипа.)
- функции как объекты первого класса. (т.е. мб сохранены в переменную, переданны в функцию как аргумент, созданы во время выполнения программы и т.п.)

## 2. Jsp action

JSP actions могут воздействовать на стандартный поток вывода, использовать, модифицировать и создавать объекты используя конструкции XML для управления сервлетом.

```
<jsp:action_name attribute = "value" />
```

- `jsp:include` - включает файл при запросе
- `jsp:useBean`
- `jsp:setProperty`
- `jsp:forward` - пересылает на другую страницу
- `jsp:declaration` – объявление, аналогично `<%! … %>`
- `jsp:scriptlet` – скриптлет, аналогично `<% … %>`
- `jsp:expression` – скриптлет, аналогично `<%= … %>`
- `jsp:text` – вывод текста
- ...

# Билет ???

## 1. Правила создания HTML-форм

Форма предназначена для обмена данными между пользователем и сервером.

Задается с помощью тега `<form>` и могут содержать в себе атрибуты: `action`, содержищий URI обработчика формы (обязательный атрибут), `method` (по умолчанию, GET), `enctype` (тип кодирования), `accept` (MIME-типы для загрузки файлов), `name`, `onsubmit/onreset` (обработчик события submit/reset для скриптов), `accept-character`.

Виды полей:

- Кнопка `<input>`. Типы кнопок `<submit>`, `<image>`, `<reset>`, `<button>`.
- Checkbox.
- Radio.
- Select.
- Text и многострочный textarea.
- Password.
- Hidden (скрытое поле).
- File.

# 2. Конфигурационный файл web.xml

Java веб-приложения используют файл дескриптора развертывания `web.xml` для определения какие URL будут передаваться определенному сервлету, какие URL требуют аутентификации. Дескриптор развертывания веб-приложений описывает классы, ресурсы и конфигурацию приложения, а так же как сервер будет использовать их для выполнения веб-запросов.

```
    <servlet>
        <servlet-name>redteam</servlet-name>
        <servlet-class>mysite.server.TeamServlet</servlet-class>
        <init-param>
            <param-name>teamColor</param-name>
            <param-value>red</param-value>
        </init-param>
    </servlet>
```

Определяет соответствие между путями URL и сервлетами

```
<servlet-mapping>
        <servlet-name>redteam</servlet-name>
        <url-pattern>/red/*</url-pattern>
</servlet-mapping>
```

# Билет ????

# Структура протокола http, характеристики

HTTP — протокол прикладного уровня передачи данных. Основой HTTP является технология «клиент-сервер».

HTTP запрос состоит из трех основных частей, которые идут в нем именно в определенном порядке:

1. строка запроса (Request Line) - указывает метод передачи, URL-адрес, к которому нужно обратиться и версию протокола HTTP.
2. заголовки (Message Headers) - описывают тело сообщений, передают различные параметры и др. сведения и информацию.
3. пустая строка (разделитель)
4. тело сообщения (Entity Body) – необязательный параметр - данные, которые передаются в запросе.

## Методы запросов:

- `GET` запрашивает представление ресурса, может только извлекать данные.
- `HEAD` как GET, только без тела ответа.
- `POST` используется для отправк сущностей определенному ресурсу. Может изменять данные.
- `PUT` создает новый ресурс или заменят предсавление целевого ресурса (в отличае от POST для идентичных наборов двнных будет иметь одинаковый результат).
- `DELETE` удаляет ресурм.
- `CONNECT` устанавливает "туннель" к серверу, определенному по ресурсу.
- `OPTIONS` для описания параметров соединения с ресурсом
- `TRACE` вызов возвращаемого текстового сообщения
- `PATCH` частичное изменение ресурса

## Коды ответов:

1. 1xx: Informational
2. 2xx: Success
3. 3xx: Redirection (перенаправление)
4. 4xx: Client Error
5. 5xx: Server Error

# 2. Жизненный цикл Jsp

Конвертацией JSP страниц в HTML код занимается контейнер.

## Жизненный цикл:

1. `Translation` – JSP контейнер проверяет код JSP страницы, парсит ее для создания кода сервлета.
2. `Compilation` – JSP контейнер компилирует исходный код jsp класса и создает класс на этой фазе.
3. `Class Loading` – контейнер загружает классы в память на этой фазе.
4. `Instantiation` – внедрение конструкторов без параметров созданных классов для инициализации в памяти классов.
5. `Initialization` – в контейнере вызывается init метод объекта JSP класса и инициализируется конфигурация сервлета с init параметрами, которые указаны в дескрипторе развертывания (web.xml).
6. `Request Processing` – длительный жизненный цикл обработки запросов клиента JSP страницей. Обработка является многопоточной и аналогична сервлетам — для каждого запроса создается новая нить, создаются объекты ServletRequest и ServletResponse и происходит внедрение сервис методов JSP.
7. `Destroy` – последняя фаза жизненного цикла JSP на которой JSP класс удаляется из памяти. Обычно это происходит при выключении сервера или андеплое приложения. Методы: `jspInit()` , `_jspService()`, `jspDestroy()`

# Не вошло в билеты, но есть в вопросах

## Вопросы

1. Стандарты и протоколы сети интернет.
2. ~~Протокол HTTP.~~
3. ~~Язык HTML.~~
4. ~~CSS (LESS, SASS, SCSS).~~
5. ~~JavaScript~~ (ES5 / ES6 / ES7).
6. ~~DHTML и AJAX,~~ jQuery, SuperAgent.
7. Long Polling и WebSockets.
8. ~~CGI, FastCGI.~~
9. ~~PHP.~~
10. ~~Версии платформы Java, Java EE, контейнеры, компоненты.~~
11. Сервлеты.

12. ~~JSP.~~
13. ~~Шаблоны проектирования и архитектурные шаблоны. Архитектура интернет-приложений.~~

# 5. ES5 / ES6 / ES7

ECMAScript — стандарт языков, не имеющий средств ввода-вывода язык программирования, используемый в качестве основы для построения других скриптовых языков.

Имеет 5 примитивных типов данных — Number, String, Boolean, Null и Undefined;
Объектный тип данных — Object и 15 различных видов инструкций.

В особенности можно добавить то, что блок не ограничивает область видимости функции. Если переменная объявляется вне функции, то она попадает в глобальную область видимости. Функция — это тоже объект.

# ES6

Это обновление добавило новый синтаксис для написания классов и модулей, добавились итераторы и циклы for/of, Python-style генераторы, двоичные данные, лямбда-выражения, типизированные массивы, коллекции, обещания (promises), рефлексию и прокси, усовершенствовали числа и математику. Добавлено ключевое слово let (которое помогает объявить переменной область видимости - блок) и const.

# ES7

Добавлена операция возведения в степень (**), Array.prototype.includes().

# 6. JQuery

`jQuery` - библиотека js, помогающая легко получить доступ к любому элементу DOM и манипулировать ими, предоставляет API для работы с AJAX.

jQuery включается в страницу как внешний файл.

```
<script src="jquery-2.2.2.min.js">
```

Вся работа с jQuery ведется с помощью функции `$`. Работу с jQuery можно разделить на 2 типа:

- Получение jQuery-объекта с помощью функции `$()`.

- Вызов глобальных методов у объекта `$`.

Типичный пример манипуляции сразу несколькими узлами DOM заключается в вызове `$` функции со строкой селектора CSS, что возвращает объект jQuery, содержащий некоторое количество элементов HTML-страницы. Эти элементы затем обрабатываются методами jQuery.

```
$("div.test").add("p.quote").addClass("blue").slideDown("slow");
//находит элементы div с классом test, все элементы p с классом quote,
добавляет им класс blue, ...
```

`$.ajax` и соответствующие функции позволяют использовать методы AJAX

```
$.ajax({
  type: "POST",
  url: "some.php",      // обращение к some.php
  data: {name: 'John', location: 'Boston'},   //с какими-то параметрами
  success: function(msg){
    alert( "Data Saved: " + msg );  // получный результат выводится в
alert
  }
});
```

# 8. Long Polling и WebSockets.

# Regular http

1. Клиент посылает запрос на вебстраницу к серверу
2. Сервер формирует ответ
3. Сервер посылает ответ клиенту

# Polling

1. Клиент посылает запрос серверу, используя обычный http
2. Клиент запрашивает файл от сервера через какой-то интервал времени
3. Сервер формирует ответ для каждого запроса и отсылает его обратно

# Long Polling

1. Клиент запрашивает страницу у сервера, используя обычный http
2. Запрошенная страница выполняет JavaScript, который запрашивает файл от сервера.

3. Сервер НЕ реагирует на запрошенную информацию и ждет, пока не появится новой информации
4. Когда появляется новая информация, сервер отсылает ее клиенту
5. Клиент получает новую информацию и сразу отсылает другой запрос серверу, запуская процесс ожидания на нем снова.

## WebSokets

1. Клиент запрашивает страницу у сервера, используя обычный http
2. Открывается соединение с сервером
3. Сервер и клиент могут посылать друг другу сообщения

# 12. Сервлеты.

Пакеты `javax.servlet` и `javax.servlet.http` обеспечивают интерфейсы и классы для создания сервлетов.

**Сервлет** — это Java-класс, который наследуется обычно от класса `HttpServlet` и переопределяет часть методов:

- `doGet` — если мы хотим, чтобы сервлет реагировал на GET запрос.
- `doPost` — если мы хотим, чтобы сервлет реагировал на POST запрос.
- `doPut`, `doDelete` — если мы хотим, чтобы сервлет реагировал на PUT и DELETE запрос (есть и такие в HTTP). Эти методы реализуются крайне редко, т.к. сами команды тоже очень редко встречаются.
- `init`, `destroy` — для управления ресурсами в момент создания сервлета и в момент его уничтожения.

```
public class NewServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        // Параметр
        String parameter = request.getParameter("parameter");

        // Старт HTTP сессии
        HttpSession session = request.getSession(true);
        session.setAttribute("parameter", parameter);

        response.setContentType("text/html;charset=UTF-8");
```

```
        PrintWriter out = response.getWriter();
        try {
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Заголовок</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Пример сервлета"+parameter+"</h1>");
            out.println("</body>");
            out.println("</html>");
        } finally {
            out.close();
        }
    }

    @Override
    public String getServletInfo() {
        return "Пример сервлета";
    }

}
```

## Сервлет vs CGI

- Сервлеты запускаются в одном процессе (HTTP-сервер с дополнительными функциями, который называется Serlet Container), и они существуют до тех пор, пока этот процесс существует.
- CGI каждый раз создает новый экземпляр процесса для обслуживания запроса. Это убийца перфоманса.
- Поскольку для каждого запроса существует новый процесс, это означает, что CGI не может агрегировать данные из нескольких запросов в памяти.

## Сервлет vs FastCGI

- При использовании сервлетов веб-сервер может напрямую вызвать приложение.

# ПРАКТИЧЕСКИЕ ЗАДАНИЯ

## Задания на JS

Написать функцию, которая на странице заменяет все текстовые поля ввода на кнопки:

```
function replaceTextFields() {

Array.prototype.slice.call(document.querySelectorAll('input[type=text]')
).forEach(function(textField) {
    textField.setAttribute('type', 'submit');
  });
}
```

Написать функцию, которая запрещает писать числа и буквы латинского алфавита:

```
document.querySelector('#textfield').onkeypress = function(e) {
  var input = e.key.toLowerCase();
  if (input ⩾ 'a' && input ⩽ 'z') return false;
  if (input ⩾ '0' && input ⩽ '9') return false;
  return true;
};
```

Написать функцию, которая заменяет содержимое `<div>` с именем класса `nyan` на изображение по ссылке: http://www.example.com/nyancat.gif:

```
function insertNyancat() {
  document.querySelector('div.nyan').innerHTML = '<img
src="http://www.example.com/nyancat.gif">';
}
```

Написать функцию, которая будет закрывать текущее окно, если в нем открыт https://www.google.ru:

```
function leaveIfGoogle() {
  if (window.location.href.startsWith('https://www.google.ru'))
    window.close();
}
```

JS-функция, открывающая в новом окне браузера сайт http://www.google.com

```
window.open("http://google.com", "newwin", "width=1200,height=600");
```

Реализовать функцию на JavaScript, которая будет закрывать текущее окно, если в нем открыт https://www.google.ru

```
function close_window() {
  if (window.location.href===”https://www.google.ru”) {
        window.close();
  }
}
```

JS-функция, удаляющая со страницы все элементы

с классом “MarkedToRemove”

```
Array.prototype.slice.call(document.getElementsByTagName("div")).forEach
(function (e) {
   if (e.getAttribute("class") === "MarkedToRemove"){
       e.parentNode.removeChild(e);
   }
});
```

# Задания с CSS

Правило css, меняющее цвет фона на желтый, если ссылка посещена и не лежит в классе "news"

```
a:visited:not([class*="news"]) {
        background: yellow;
}
```

Написать css правило, которое при клике на ссылку добавляет ей подчеркивание, всем кроме ссылок в теге h1

Тупо, но работает!

```
      a:active{
        text-decoration: underline;
      }

      h1 > a:active{
        text-decoration: none;
      }
```

css правило, выравнивающее все блоки

внутри формы с id=”sampleForm” по правому краю

```
#sampleForm div{
text-align:right
}
```

# Задания по сервлетам

Написать сервлет, который принимает из http запроса параметр name и выводит его. Если параметр не обнаружен то вывести Anonymous user

```
public class NameServlet extends HttpServlet {
  public void service(HttpServletRequest request,HttpServletResponse
response) throws IOException, ServletException {
    PrintWriter out = response.getWriter();
    if (request.getParameter("name") ≠ null ) {
      out.println(request.getParameter("name"));
    } else {
      out.println("Anonymous user");
    }
  }
}
```

Код фильтра запросов, запрещающий доступ к приложению неавторизированным пользователям(у неавт пол в запросе отсутствует заголовок x-application-user

```
public class TestFilter implements Filter {

    @Override
    public void init(FilterConfig filterConfig) throws ServletException
{
    }

    @Override
    public void doFilter(ServletRequest servletRequest, ServletResponse
servletResponse, FilterChain filterChain) throws IOException,
ServletException {

        HttpServletRequest httpServletRequest = (HttpServletRequest)
servletRequest;

        if(httpServletRequest.getHeader("x-application-user") == null){
```

```
httpServletRequest.getRequestDispatcher("/").forward(servletRequest,
servletResponse);
        }

        filterChain.doFilter(servletRequest, servletResponse);
    }

    @Override
    public void destroy() {

    }
}
```

Написать сервлет, который будет возвращать все запросы на переадресацию на сайт google.com

```
public class RedirectServlet extends HttpServlet {
        RedirectFilter filter = new RedirectFilter;
        public void service(HttpServletRequest request,
HttpServletResponse response) throws IOExeption, ServletException {
                filter.doFilter(request, response);
                int status = response.getStatus();
                if (status == 302) {
                        response.sendRedirect("http://google.com");
                }
        }
}
```

# JSP

Страница JSP, проверяющая есть ли /какой-то параметр/ в запросе и если нету выводящая сообщение об ошибке

```
 <%
if(request.getAttribute(“123”)==null){
        String error = "Please set '123' param";
}
%>
<p>${error}</p>
```