**Ajax** – методология построения интерактивного пользовательского интерфейса web-приложения.

**Методология** **Ajax описывает** способы разработки страниц сайта, которые могут динамически изменяться на основе данных поступающих с сервера, но без полной перезагрузки страницы.

**В основе методологии Ajax лежат следующие технологии:**  язык HTML, язык JavaScript, язык XML, модель DOM, протокол HTTP, протокол JSON, объект XMLHttpRequest.

**HTML** **–**гипертекстовый язык разметки.

**JavaScript** – скриптовый язык, предназначенный для создания сценариев поведения браузера.

**DOM –** объектная модель, позволяющая сценариям JavaScript получить доступ (читать и изменять содержимое) к элементам html-документа (к атрибутам и содержимому тегов). В Ajax ответ сервера “встраивается” с помощью JavaScript-сценария в загруженную ранее браузером страницу. При этом доступ к элементам html-документа осуществляется а соответствии с моделью DOM.

**HTTP –** сетевой протокол передачи гипертекста. Используется для обмена данными между двумя приложениями (клиентом и сервером).

**XML –** расширяемый язык разметки данных. Предназначен для структуризации данных с целью хранения или/и передачи. В Ajax язык XML является одним из форматов, который используется для структуризации данных пересылаемых между JS-сценарием и серверным приложением.

**JSON (JavaScript Object Notation)**  - текстовый формат обмена данными, применяемый обычно в JavaScript**.** В Ajax формат JSON используется для структуризации данных пересылаемых между JavaScript-сценарием и серверным приложением. Формат JSON основывается на функции **eval()** языка JavaScript.

**XMLHttpRequest** –специальный API, используемый в JS для обмена данными между сценарием на JS и серверным приложением по протоколу HTTP. В Ajax методы объекта XMLHttpRequestиспользуется для отправки и получения данных между JavaScript-сценарием и серверным приложением. Данные могут получены в виде XML-документа и виде обыкновенного текста.

**Задание 1. Применение заголовков для обмена между сценарием JavaScript серверным приложением.**

1. Разработайте сервлет **Sss\_Header**, который принимает два заголовка **Value-x** и **Value-y**, значения которых целые числа **x** и **y**. Сервлет должен сложить два числа и сформировать заголовок ответа **Value-z,** значение которого сумма **x+y**. Используйте пример 1.
2. Убедитесь в работоспособности сервлета **Sss\_Header**.
3. Разработайте форму для ввода значений **x**, **y** и **z**. Разработайте сценарий JavaScript, который упаковывает **x** и **y**  в заголовки с именем **Value-x** и **Value-y** в http-запрос и оправляет его сервлету **Sss\_Header,** а после этого принимает ответ, распаковывает заголовок ответа **Value-z** и помещает его в тег, соответствующий **z**. Используйте примеры 2-4

**Задание 2. Применение XML для обмена между сценарием JavaScript серверным приложением.**

1. Разработайте сервлет **Sss\_XML**, который принимает заголовок **XRand-N**, значение которого целое число **n**. Сервлет должен сформировать ряд целых случайных чисел из интервала **(-n, n)**. Количество сформированных чисел в ряду должно быть случайным из интервала **[5,10]**. Полученный ряд необходимо отправить клиенту в виде xml-документа.
2. Убедитесь в работоспособности сервлета **Sss\_XML**.
3. Разработайте форму для ввода значений **n** и вывода ответа. Разработайте сценарий JavaScript, который упаковывает значение **n**  в заголовок с именем **XRand-N** http-запроса и оправляет его сервлету **Sss\_XML,** а после этого принимает ответ, разбирает полученный xml-документ в тело тега, предназначенного для отображения результата.

Используйте примеры 1,3,5,6,7.

**Задание 3. Применение JSON для обмена между сценарием JavaScript серверным приложением.**

1. Разработайте сервлет **Sss\_JSON**, который принимает заголовок **XRand-N**, значение которого целое число **n**. Сервлет должен сформировать ряд целых случайных чисел из интервала **(-n, n)**. Количество сформированных чисел в ряду должно быть случайным из интервала **[5,10]**. Полученный ряд необходимо отправить клиенту в формате JSON.
2. Убедитесь в работоспособности сервлета **Sss\_JSON**.
3. Разработайте форму для ввода значений **n** и вывода ответа. Разработайте сценарий JavaScript, который упаковывает значение **n**  в заголовок с именем **XRand-N** http-запроса и оправляет его сервлету **Sss\_JSON,** а после этого принимает ответ, разбирает полученный json-текст в тело тега, предназначенного для отображения результата.
4. Используйте примеры 1,3,5,6,8,9.

**Задание\_4. Применение асинхронной обработки.**

1. Сведите все три формы, сделанные в заданиях 1-3 в одну.
2. Установите в вызове метода **open** объекта **XMLHTTPRequest** значение третьего параметра (асинхронность) в **false**.
3. Внесите изменения в разработанные сервлеты таким образом, чтобы они выполняли http-ответ с задержкой: Sss\_Header(10 сек), Sss\_XML (5 сек), Sss\_JSON (1 сек.) Используйте пример 11.
4. Выполните приложение и объясните эффект.
5. Верните значение параметра в методе **open** в состояние **true**.
6. Проверьте возможность асинхронной работы вашего приложения (не дожидаясь результата в первой форме, сразу выполнять вторую и третью). Объясните полученный результат. **Исправьте ошибку (если она есть)**.