**Современные стандарт асинхронного взаимодействие на js**

Объекты promise

Новый js объект, который предоставляет асинхронную операцию решающий проблемы с

Разработки интерфейса и coolback. За счет того что что делает асинхронный код похожим на синхронный

**Promise API**

при создание объекта типа promise в конструткор нужно передать

функцию coolback выполняющий асинхронную операцию

excuter - исполнитесь

должен принять coolback 1-resolve 2-reject

resolve - в случаи успеха операции

reject - в случаи неудачи

при успешном выполнении асинхронной операции если ответ имеется

он будет передан resolve если ошибка то в reject

excutor выполняется синхронно но выполняет он асинхронную операцию значит

что она может вернуть управление до завершения асинхронной операции

**возможные состояния объекта promise**

1. fulFillect – выполнено если вызвана функция resolve с аргументом не являющемся с аргументом объектом promise или без аргумента
2. reject – отклонено если вызвана f reject или возникло исключение внутри excutora
3. pending – ожидание если не resolve, reject еще не вызван
4. settled – завершено при завершении операции выполнена она или отклонена

ДЗ bubling cuption в DOM всплытие this target element

**Fetch API   
async await**

F async в случ когда вы пишете return возвращает промис в состояние resolve, инструкция await ожидает состояние settl переданным промисом , для обработки ошибок в async используется try catch

**Модули es6**

Это пере использованная часть кода, содержит в себе детали реализации и предоставляет открытое api, что позволяет легко загрузить её и использовать в другом коде

Модули в js позволят нам:

1. абстрагировать код (означает передавать функциональный возможности внешним библиотекам так что нам не придется разбираться во всем сложностях реализации)
2. инкапсулировать код (позволяет скрывать код внутри модуля если мы не хотим чтобы его изменяли )
3. позволяют управлять зависимостями

форматы модулей

это синтаксис который используется для его определения

широко использованные форматы модулей:

1. асинхронное определение модулей (AMD)
2. commons
3. универсальное определение модуля (UMD)

асинхронное определение модуля AMD

формат AMD используется в браузерах и применяет для определения функцию defined

commonJS

применяется в node Js и использует для определения зависимости и модулей функцию requirejs и модул експартс

формат UMD

может использоваться как у браузере так и в node JS

загрузчики модулей

интерпретирует и загружает модулю написанный в определенном формате

загрузчик модулей

1. вы загружаете в браузере
2. вы сообщаете загрузчику какой main файл запустить
3. модулю скачивает и интерпретирует главный файл приложения
4. скачивает файлы по мере необходимости

существует несколько популярных загрузчиков

1. requirejs в формате AMD
2. систем js формат AMD, common js, UMd

сборщики модулей

заменяет собой загрузчика однако в отличии сборщик модулей запускается при сборке

1. вы запускаете сборщик модулей для создания файла пакетов во время сборки bantel js файл пакеты

популярные сборщики – Browserify (для common js), webpack (для AMD es6)

дз AMD, UMD, common js, CORS