1. **Внешние скрипты, порядок выполнения скриптов.**

* Что?

Скрипты, код которых располагается либо в отдельном js файле, либо удаленно.

* Зачем?

Используются для того, чтобы не перегружать сайт и дает возможность использовать одинаковые скрипты на разных страницах сайта. Браузер подгружает такие скрипты всего один раз, а в дальнейшем использует необходимые данные уже из своего кэша. Таким образом нет нужны делать много запросов на сервер, что значительно увеличивает скорость работы сайта.

* Как работает?

Обычно такие скрипты подключаются с помощью тега script и атрибута src, пишется перед закрывающим тегом </body>. Нельзя использовать код внутри тэга и атрибут src одновременно.

Скрипты подгружаются последовательно, и если бы мы указывали их в середине документа, то приходилось бы ждать пока выполнится скрипт, и только потом видели бы остатки хтмл кода под ним. Именно по этой причине скрипты рекомендуется подключать после всего контента на странице, но это не решит проблему полностью.

Некоторые скрипты тоже могут тупить, и скрипты под ним будут вынуждены ждать своей очереди, что тоже не есть хорошо. Для решения этой проблемы существует для атрибута:

**Async** – откладывает исполнение скрипта до его полной загрузки. Т.е. браузер считывает этот атрибут и понимает, что можно двигаться дальше, прогружая другой контент, а этот скрипт исполнить тогда, когда он прогрузится. При этом последовательность скриптов не сохраняется и первым выполнится тот, кто меньше весит (например), ну или по каким-то другим причинами загрузится быстрее. Не работает в ишаке до девятой версии. Не ждет, пока загрузится вся страница.

**Defer** –Работает почти также как асинк, но поддерживается всеми браузерами, соблюдается порядок исполнения скриптов. Даже если скрипт прогрузился раньше, он не будет исполнен, если скрипт над ним еще не прогрузился. Этот атрибут используется тогда, когда скрипты зависят друг от друга. Например, скрипт с jquery и все остальные скрипты. Ждет пока загрузится вся страница.

Данные атрибуты работают только для внешних скриптов, которые имеют атрибут src.

**2 Структура кода**

* Что?

Свод правил, которых необходимо придерживаться, программируя на языке javascript

* Зачем?

Для того, чтобы избежать ошибок, которые трудно будет найти, а также для соблюдения общих правил и стандартов для всех разработчиков в больших проектах плюс для того, чтобы код лучше воспринимался на глаз и был более читабельным.

* Как работает?

1. Необходимо ставить точку с запятой после каждой команды
2. // Однострочный комментарии
3. /\*Многострочный комментарий \*/
4. В комментариях не допускается вложенность

**3 Особенности стандарта ES5, директива 'use strict'**

* Что?

Как я поняла, директива юз стрикт это такая фишка, которая позволяет писать код используя современный стандарт языка экмаскрипт 5.

* Зачем?

Чтобы быть современным :D Фиг знает конкретно ЗАЧЕМ.

* Как работает?

В начале кода, если мы решаемся исползовать ес5, нам необходимо написать строчку «use strict»; После этого наш код будет работать в рамках экмаскрипт 5, но не факт, что наш старый код сможет работать вместе с новым кодом.

Мне тут сильно ниче не понятно, просто поняла, что есть строгий режим написания кода, который не поддерживается даже 9м ишаком, ему надо подключать сторонние скрипты, не говоря уже о более старых версиях.

**4 Переменные**

* Что?

Область памяти, которая имеет своё имя.

* Зачем?

Для хранения и дальнейшего использования определенного вида информации.

* Как работает?

Переменная создается с помощью команды var. С помощью = присваивается значение, которое будет храниться в переменной до тех пор, пока она не начнет использоваться и видоизменяться.

Имена переменных в жс регистрозависимые. Также существует определенный список символов, использование которых в именах переменных недопустимо. Русские буквы также не приветствуются. В режиме юз стрикт нельзя объявлять переменную без var.

Константы – переменные которые не меняются. Принято называть заглавным буквами через нежнее подчеркивание (если содержит 2 слова). Используют вместо трок и цифр, чтобы не допускать опечаток и сделать код понятнее.

**5 Правила выбора имени переменной**

* Никакого транслита, только английские слова.
* Называть переменные очень короткими именами допустимо только там, где их применение очевидно, и они расположены в небольшом фрагменте.
* Переменные из нескольких слов пишутся верблюжьей нотацией либо разделяя слова нижним подчеркиванием
* Имя переменной должно максимально четко отображать сущность и характер данных, которые в ней хранятся.
* Имена должны быть максимально понятными, чтобы в любой момент можно было найти любую нужную переменную в кратчайшие сроки.
* Если есть необходимость в переменной, то нужно ее создавать, а не пытаться использовать уже имеющуюся. Дополнительная переменная в большинстве случаев – хорошо, а не плохо.

**6 Типы данных, type of**

* Что?

Все данные можно очень четко разделить на 6 видов.

* Зачем?

Для того, чтобы знать с каким типом данных ты работаешь :D

* Как работает?

1 тип: число, также сюда входят два специальных значения infinity – бесконечность

И NAN – ошибка вычислений.