ОСНОВЫ JAVASCRIPT

26.06.19

GIT

- GIT – одна из систем контроля версий. Обеспечивает Разработчикам параллельную правку кода. Представляет из себя дерево веток проекта
- Master – ключевая ветка проекта, от неё всё порождается
- Git checkout {ветка} – переключение на ветку. Пример: git checkout master, git checkout feature/HS-32155 (привел пример как в жизни именуются ветки).
- Git checkout –b feature/HS-35554 (переключение на ветку. Если её не найдено, то будет создана)
- Git pull {ветка} – стянуть последние изменения с ветки. Если ветку не указывать, то будут стянуты изменения в ветке, в которой вы сейчас находитесь
- Git status – проверка текущего состояния ветки. Покажет, какие несохраненные изменения есть по сравнению с загруженной версией ветки, какие коммиты уже есть в ветке

GIT

- Git add [path1, path2, …] – добавление файлов, которые идут в коммит
- Commit – логически выполненный кусок задачи. Пример: задача – реализовать график на какой-нибудь странице. Т.е. первый commit – ищем, какую технологию лучше для этого использовать и вот мы её подключили, второй – сделать, чтоб график строился на основе наших данных. Третий – подровнять цвета и другие тонкости, ну а дальше – всякие баги, если такие обнаружатся
- Git commit –m “здесь кратко пишем, что сделали. Парой фраз.”
- Git push {branch} – пушим наши commit-ы в ветку, которую мы указали. Если не указывать ветку – пушится в текущую
- Git merge – порой бывает, что мы работаем с каким-то файлом, над которым уже успел потрудиться другой разработчик и запушить в ту же ветку, куда мы хотим загрузить свои изменения. Но он сделал это первый, значит Git будет полагать, что этот файл нуждается в ручном принятии нужных изменений. Git подсказывает, какие именно изменения конфликтные

JS FUNCTIONS

```
function имя_функции {
        var a = +prompt(“Введите число”);
        var answer = ‘’;
        if (a > 0) {
                answer = ‘положительное’;
                //…….. Обычные инструкции
        }
        return answer;
}
```

JS FUNCTIONS

То, что находится внутри функции – называется Тело функции

То, что возвращает функция – return. После return функция прекращает работу и захлопывается, передавая дальше в код результат (если нужно)

```
function add(x, y, z) {
        return x + y + z;
}
var answer = add(1, 3, 5); //в answer попадет 9
```

JS УСЛОВНЫЕ ОПЕРАТОРЫ

```
If (условие1) {
        Тогда нам сюда
} else if (условие2) {
        Если не условие1 но попытаемся еще раз, спрашиваем условие2
} else {
        Выполнится, если с условиями выше совсем печаль
}
```

Все условия приводятся к Boolean ответу: либо true, либо false

Если хотим спросить о равенстве в условии, то это либо

== (браузеру разрешается попробовать привести данные к одному типу), либо

=== (тут уже если данные разных типов (number и string, к примеру), то всегда false)

JS УСЛОВНЫЕ ОПЕРАТОРЫ

JS приводит типы

1 == ”1” //true

0 < ‘5’ //true, встречает больше/меньше/больше-равно/меньше-равно – сразу обрабатывает как числа

0 < ‘-1’ // false

true == {} //false

false == {} //false

1 == true //true

1.05 == true //false

2/3 == true //false

2/3 == false //false