Git. Настройка среды. Преобразование типов. Операторы сравнения

Занятие 2


O'REILLY
Изучаем JavaScript
Давай. Вошли и вышли, приключение на 20 минут.

Ход занятия

- git
- Настройка среды
- Преобразование типов
- Операторы сравнения

Git

Git — распределённая система контроля версий, которая даёт возможность разработчикам отслеживать изменения в файлах и работать над одним проектом совместно с коллегами. Она была разработана в 2005 году Линусом Торвальдсом, создателем Linux, чтобы другие разработчики могли вносить свой вклад в ядро Linux. Git известен своей скоростью, простым дизайном, поддержкой нелинейной разработки, полной децентрализацией и возможностью эффективно работать с большими проектами.

Подход Git к хранению данных похож на набор снимков миниатюрной файловой системы. Каждый раз, когда вы сохраняете состояние своего проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок.

GitHub

GitHub — сервис онлайн-хостинга репозиториев, обладающий всеми функциями распределенного контроля версий и функциональностью управления исходным кодом — всё, что поддерживает Git и даже больше.

Git-репозиторий, загруженный на GitHub, доступен с помощью интерфейса командной строки Git и Git-команд. Также есть и другие функции: документация, запросы на принятие изменений (pull requests), история коммитов, интеграция со множеством популярных сервисов, email-уведомления, эмодзи, графики, вложенные списки задач, система @упоминаний, похожая на ту, что в Twitter, и т.д.

Установка git

Установка на windows:

1. Скачать и установить по ссылке git для windows. Ссылка
2. Проверяем установку в командной строке windows вводим команду: git --version
3. Подключаем git к терминалу VSCode, в командной строке выполняем команду: git config --global core.editor "code --wait"
4. Проверяем установку в терминале VSCode вводим команду: git --version

Установка на ubuntu:

1. В терминале VSCode выполнить команду: sudo apt install git
2. Проверяем установку в терминале VSCode вводим команду: git --version

Настройка git

Перед началом работы необходимо установить имя пользователя и email, который будет отображаться в гите. Для этого следует выполнить следующие команды:

git config --global user.name "John Doe"

git config --global user.email johndoe@example.com

Ссылка на документацию

Инициализация репозитория

Следуя инструкциям выполняем инициализацию репозитория. Пользоваться будем HTTP версией, SHH можно настроить по желанию (гайд по настройке ssh)

a. echo "# repo_name" >> README.md - создаем файл README.md и записываем в него строку "# repo_name"
b. git init - производится инициализация репозитория
c. git add README.md - добавляем файл в отслеживаемые
d. git commit -m "first commit" - текст коммита
e. git branch -M main - создаем локальную ветку main
f. git remote add origin https://github.com/user_name/repo_name.git - указываем ссылку на удаленый репозиторий
g. git push -u origin main - отправляем изменения в удаленную ветку main

Настройка среды

Node JS

Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, написанный на C++, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода.

Установка Node JS

Установка на windows:

1. Скачать и установить по ссылке node js для windows. Ссылка
2. Проверяем установку в командной строке windows вводим команду: node -v
   (перед запуском команды перезапустите VSCode)

Установка на ubuntu:

1. Устанавливаем node js по любому гайду
2. Проверяем установку в терминале вводим команду: node -v

Запуск скрипта

Для того, чтобы запустить выполнение кода в среде node js. Необходимо в терминале ввести команду: node относительный_адрес_файла

После выполнения команды, мы запустим скрипт и результат работы увидим в терминале (если в самом скрипте присутствовали console.log())

Prettier

Prettier - это средство для форматирования кода и поддержания стиля написания кода по жестко заданным пользователем правилам правилам.

Установить prettier можно скачав расширение для VSCode или выполнив следующие действия:

1. Нажать сочетание клавиш Ctrl+P и в открывшемся окне выполнить команду: ext install esbenp.prettier-vscode
2. После установки, выбрать файл, нажать правой кнопкой мыши и выбрать пункт (форматировать документ с...) или (format document with...). В открывшемся окне выбрать: prettier - code formatter

Итог

После правильной установки и настройки рабочего окружения у нас должны быть:

1. Заведен аккаунт GitHub и настроен git в системе
2. Установлена последняя LTS версия Node JS
3. Настроен Prettier и VSCode
4. Заведен учебный репозиторий для работы и домашних заданий

Часть №2

...

Ход занятия

- Повторение предыдущих лекций
- Преобразование типов
- Операторы сравнения

Переменные

Какие имена из приведенного ниже списка не могут быть использованы в JavaScript и почему?

- $$userName
- course-count
- 52Region
- is_user_from_russia
- thisTime
- new
- _currentValue
- #courseDescription
- a

Переменные

В чем принципиальное отличие var от let и const?
Что будет в результате выполнения данного кода?

```
const isAdmin = true;

isAdmin = false;
```

Типы данных

Сколько типов данных существует в JavaScript?

Что будет в результате выполнения данного кода?

```
console.log(typeof('true'));

console.log(typeof('Ivan'/4));

const obj = {

        foo: "bar",

        };

console.log(typeof(obj));
```

Преобразование типов

Преобразование типов

Преобразование типов - это процесс конвертации значения из одного типа в другой (как например, строки в число, объекта к булевому значению и т. д.)

Можем выделить три вида преобразования:

- Строковое преобразование
- Численное преобразование
- Логическое преобразование

Строковое преобразование

Строковое преобразование – происходит, когда нам нужно что-то вывести в виде строки. Может быть вызвано с помощью String(value). Для примитивных значений работает очевидным образом:

- true становится “true”
- 45 становится “45”
- null становится “null”

Пример вызова: typeof String(38) === “string”; // значение “38”

Численное преобразование

Численное преобразование – происходит в математических операциях. Может быть вызвано с помощью Number(value). Если строка не может быть явно приведена к числу, то результатом преобразования будет NaN. Работает по следующим правилам:

- undefined становится NaN
- null становится 0
- true / false становится 1 / 0
- string преобразуется по правилу, пробельные символы по краям обрезаются. Далее, если остаётся пустая строка, то получаем 0, иначе из непустой строки «считывается» число. При ошибке результат NaN.

Пример вызова: typeof Number(“”123zxc”) === “number”; // значение NaN

Логическое преобразование

Логическое – Происходит в логических операциях. Может быть вызвано с помощью Boolean(value).
Главное запомнить, что если строка не пустая, то она всегда true.
Работает по следующим правилам:

- 0, null, undefined, NaN, “” становится false
- любое другое значение становится true

Пример вызова: typeof Boolean(“Hello World”) === “boolean”; // значение true

Исключения

Ниже приведены исключения которые надо запомнить:

- undefined при численном преобразовании становится NaN, не 0
- "0" и строки из одних пробелов типа " " при логическом преобразовании всегда true

Математические операторы

В JavaScript поддерживаются следующие математические операторы:

Поддерживаются следующие математические операторы:

- Сложение +,
- Вычитание -,
- Умножение *,
- Деление /,
- Взятие остатка от деления %,
- Возведение в степень **.

Пример: console.log(7 % 4); // результат 3

Унарный плюс

Унарный, то есть примененный к одному значению, плюс “+” ничего не делает с числами. Но если операнд не число, унарный плюс преобразует его в число.

Пример:

- console.log(typeof +5); // тип number, значение 5
- console.log(typeof +”Hello World”); // тип number, значение NaN
- console.log(typeof +false); // тип number, значение 0

Унарный плюс - это то же самое, что и Number(...), только короче.

Бинарный плюс

Бинарный, то есть примененный к двум значениям, плюс “+” складывает числа с числами. Но если хотя бы один операнд строка, то бинарный плюс объединит их в одну (конкатенация строк).

Пример:

- console.log(typeof (2 + 5)); // тип number, значение 7
- console.log(typeof (”Hello” + “ ” + “World”); // тип string, значение Hello World
- console.log(typeof (1 + “000”)); // тип string, значение 1000

Значение	Преобразование в:			
	Строку	Число	Булево	Объект
undefined null	"undefined" "null"	NaN 0	false false	ошибка typeError ошибка typeError
true false	"true" "false"	1 0		new Boolean(true) new Boolean(false)
"" (пустая строка) "1.2" "one" "-10" "+10" "011" "0xff"		0 1.2 NaN -10 10 11 255	false true true true true true true	new String("") new String("1.2") new String("one") new String("-10") new String("+10") new String("011") new String("0xff")
0 -0 NaN Infinity -Infinity 3	"0" "0" "NaN" "Infinity" "-Infinity" "3"		false false false true true true	new Number(0) new Number(-0) new Number(NaN) new Number(Infinity) new Number(-Infinity) new Number(3)
{} (любой объект) [] (пустой массив) [9] (1 числовой элемент) arr (любой другой массив) function(){} (любая функция)	см. Преобразование объектов "" "9" см. Преобразование объектов см. Преобразование объектов	см. Преобразование объектов 0 9 NaN NaN	true true true true true	

Операторы сравнения

Операторы сравнения

В JavaScript они записываются так:

- Больше/меньше: a > b, a < b.
- Больше/меньше или равно: a >= b, a <= b.
- Равно: a == b. Обратите внимание, для сравнения используется двойной знак равенства == Один знак равенства a = b означал бы присваивание
- Не равно. В математике обозначается символом ≠,но в JavaScript записывается как a != b.

При сравнении значений разных типов JavaScript приводит каждое из них к числу.

Операторы сравнения

Все операторы сравнения возвращают значение логического типа:

- true – означает «да», «верно», «истина»
- false – означает «нет», «неверно», «ложь»

Пример:

- console.log(false == 0); // true
- console.log(typeof 5 == “string”); // false

Сравнение строк

Сравнение строк происходит по следующим правилам:

- Сначала сравниваются первые символы строк
- Если первый символ первой строки больше (меньше), чем первый символ второй, то первая строка больше (меньше) второй. Сравнение завершено
- Если первые символы равны, то таким же образом сравниваются уже вторые символы строк
- Сравнение продолжается, пока не закончится одна из строк
- Если обе строки заканчиваются одновременно, то они равны. Иначе, большей считается более длинная строка.ны. Иначе, большей считается более длинная строка

Пример:

- console.log(“авто” == “Авто”); // false
- console.log(“авто” != “автомобиль”); // true

Сравнение разных типов

Так как JavaScript это слабо типизированный язык, преобразование между разными типами может происходить автоматически, и это называется неявным преобразованием типов. И в JavaScript возможна следующая ситуация:

- Два значения равны между собой
- Но одно из них true как логическое значение, другое – false.

Пример:

```
let a = 0;

console.log( Boolean(a) ); // false

let b = "0";

console.log( Boolean(b) ); // true

console.log(a == b); // true
```

Оператор строгого равенства

Оператор строгого равенства === проверяет равенство без приведения типов.

- Если a и b имеют разные типы, то проверка a === b немедленно возвращает false без попытки их преобразования.
- Оператор строгого равенства дольше писать, но он делает код более очевидным и оставляет меньше места для ошибок

Сравнение с null и undefined

Есть особенные моменты при работе с null и undefined:

- При строгом равенстве === эти значения различны, так как различны их типы
- При нестрогом равенстве == эти значения равны друг другу и не равны никаким другим значениям. Это специальное правило языка.
- При использовании математических операторов и других операторов сравнения < > <= >= Значения null/undefined преобразуются к числам: null становится 0, а undefined – NaN.

Сравнение с null и undefined

Чтобы избежать проблем:

- Относитесь очень осторожно к любому сравнению с undefined/null, кроме случаев строгого равенства ===.
- Не используйте сравнения >= > < <= с переменными, которые могут принимать значения null/undefined, разве что вы полностью уверены в том, что делаете. Если переменная может принимать эти значения, то добавьте для них отдельные проверки.

Итог

- Просто относитесь к любому сравнению с undefined/null, кроме строгого равенства ===, с осторожностью
- Операторы сравнения возвращают значения логического типа
- Строки сравниваются посимвольно в лексикографическом порядке
- Значения разных типов при сравнении приводятся к числу. Исключением является сравнение с помощью операторов строгого равенства/неравенства.
- Значения null и undefined равны == друг другу и не равны любому другому значению
- При приведении типов null становится нулём, тогда как undefined приводится к NaN
- Будьте осторожны при использовании операторов сравнений > и < с переменными, которые могут принимать значения null/undefined. Хорошей идеей будет сделать отдельную проверку на null/undefined.

Практика

- 5 > 4
- "ананас" > "яблоко"
- "2" > "12"
- undefined == null
- undefined === null
- "" + 1 + 0
- "" - 1 + 0
- true + false
- 6 / "3"
- "2" * "3"
- 4 + 5 + "px"
- "$" + 4 + 5
- "4" - 2
- "4px" - 2
- 7 / 0
- " -9 " + 5
- " -9 " - 5
- null + 1
- undefined + 1