**12. Поток документа. Скрытие элементов. Позиционирование элементов**

В HTML формирование элементов на странице происходит сверху вниз согласно схеме документа. Слой, размещенный в самом верху кода, отобразится раньше слоя, который расположен в коде ниже. Такая логика позволяет легко прогнозировать результат вывода элементов и управлять им. Порядок вывода объектов на странице и называется **«потоком».** При этом существует несколько возможностей «вырвать» элемент из потока и придать ему почти мифические свойства. Раз он не существует в потоке, то в коде его можно описать где угодно, а также выводить в заданное место окна.

Позиционированием называется положение элемента в системе координат. Различают четыре типа позиционирования: нормальное, абсолютное, фиксированное и относительное. В зависимости от типа, который устанавливается через свойство position, изменяется и система координат.

Благодаря комбинации свойств position, left, top, right и bottom элемент можно накладывать один на другой, выводить в точке с определёнными координатами, фиксировать в указанном месте, определить положение одного элемента относительно другого и др. Подобно другим свойствам CSS управление позиционированием доступно через скрипты. Таким образом, можно динамически изменять положение элементов без перезагрузки страницы, создавая анимацию и различные эффекты.

**Нормальное позиционирование**

Если для элемента свойство position не задано или его значение static, элемент выводится в потоке документа как обычно. Иными словами, элементы отображаются на странице в том порядке, как они идут в исходном коде HTML.Свойства left, top, right, bottom если определены, игнорируются.

**Абсолютное позиционирование**

При абсолютном позиционировании элемент не существует в потоке документа и его положение задаётся относительно краёв браузера. Задать этот тип можно через значение absolute свойства position. Координаты указываются относительно краёв окна браузера, называемого «видимой областью».

Рассмотрим значение fixed, которое выводит элемент из потока и привязывает к определённой точке окна. Особенностью fixed является фиксация слоя на одном месте, причём это положение не меняется при изменении размеров окна или прокрутке страницы. Такая особенность позволяет создавать неподвижные элементы интерфейса вроде кнопки «Оставить отзыв», как показано в примере 3.10. Добавления fixed недостаточно, нужно также задать положение элемента с помощью одного или двух свойств left, top, right, bottom, они управляют положением относительно окна браузера. Также при использовании свойства position ширина слоя автоматически приравнивается ширине контента плюс, как обычно, ширина отступов, границ и полей.

**Relative**

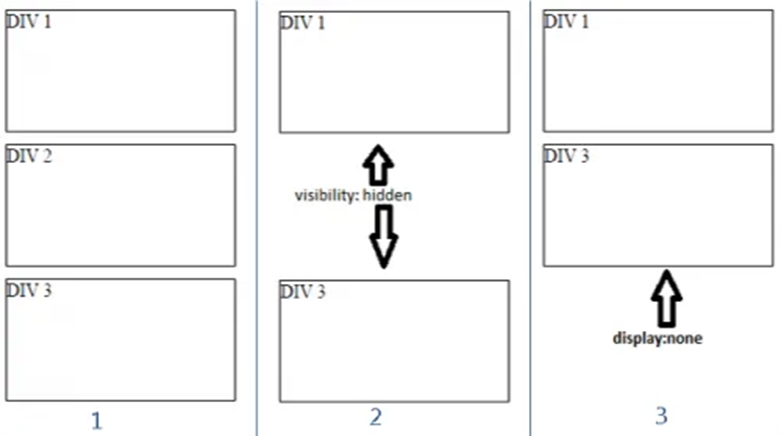
Одно из самых часто используемых значений позиционирования в CSS. Использование данного значения задает не полное, а именно относительное позиционирование относительно внешнего блока. Это означает что его границы могут изменяться с помощью свойства <padding>.Элемент может передвигаться по всей площади родительского блока, но не теряя собственных границ. Они не изменяются, сохраняя ранее использованное пространство. Оно может использоваться для других элементов.Кроме того, такой тип позиционирования позволяет блоку располагаться выше других элементов, перекрывая их. Ещё одним плюсом использования такого типа позиционирования является возможность использования свойства z-index. За счет этого относительное позиционирование в css лучше всего подходит для слоевой верстки. Данное свойство прекрасно подойдет для позиционирования картинок, кнопок, таблиц и других блоков. Для указания конкретной позиции, на которую сдвигается элемент, применяются те же свойства top, left, right, bottom.

**Скрытие элементов**

Изменить видимость элемента можно тремя способами:

1. С помощью display: none;. Элемент полностью исчезает с веб-страницы, не оставляя и следа.
2. Задав для элемента visibility: hidden;. Данный способ убирает элемент из вида, при этом не затрагивая поток html-документа, оставляя пустое пространство в том месте, где данный элемент должен был находиться.
3. С помощью свойства opacity. Значение opacity: 0; делает прозрачным весь элемент целиком, включая его содержимое, а не только фон элемента. Данный способ действует аналогично с visibility: hidden, отличие состоит лишь в том, что свойство opacity введено в спецификацию CSS3.

Для абсолютно позиционированных элементов visibility: hidden и display: none ведут себя одинаково. Свойства HTML display и visibility используются, чтобы вывести или скрыть HTML-элементы. Но тогда какая между ними разница?



Мы используем три элемента div, которые показаны в разделе 1 (с левой стороны). Изначально все элементы div выводятся без применения свойств display или visibility.

Во второй части рисунка отображаются только div 1 и div 3. Для div 2 задано visibility: hidden, но мы все равно видим пространство, которое занимает этот элемент.

В третьем разделе также отображаются только div 1 и div 3, в то время как div 2 скрыт с помощью свойства HTML display none. Можно увидеть разницу: div 3 смещается вверх и для div 2 больше не отводится пространство.

**13. Возможности инспектора элементов в браузерах, FireBug/WebInspector**

**•** Инспектирование и редактирование динамически изменяемого HTML;

• Редактирование CSS на лету;

• Отладка JavaScript, командная строка для выполнения скриптов;

• Мониторинг сетевых запросов — можно увидеть размеры и время загрузки файлов и скриптов, заголовки запросов;

**Firebug** — отладчик web-приложений, используется как отдельное расширение для браузера Mozilla Firefox, являющееся консолью, отладчиком, и DOM-инспектором JavaScript, DHTML, CSS, XMLHttpRequest. Firebug показывает в консоли вызвавшую ошибку функцию, стек вызовов функций, вызвавших эту ошибку.

**Web Inspector** - это облачный сервис, который инспектирует ваш сайт на наличие вредоносных программ, выявляет любые уязвимости и защищает ваш сайт от тысячи угроз безопасности ежедневно! Сканирование сайта с помощью Web Inspector позволяет защитить сайт от всех известных вирусов и вредоносных программ. Ваши клиенты и ваш бизнес будут ограждены от хакеров, которые могут использовать ваш сайт для заражения посетителей и нанесения ущерба.

Доверие в сети и репутация очень важны для успеха вашего бизнеса. Эта возможность позволяет ежедневно проводить исчерпывающую проверку, чтобы гарантировать, что сайт не внесен в черный список большинством поисковых систем, таких как Google и Yahoo. Инструмент Web Inspector разработан с учетом этого подхода. При обнаружении какой-либо вредоносной программы или уязвимости он немедленно отправляет уведомление на ваш email, чтобы вы могли сохранить свой сайт в безопасности. Web Inspector - инструмент для защиты сайтов, который имеет отдельную консоль для сканирования уязвимостей и проверки PCI-соответствия. Это очень важная возможность, которая поможет вам избежать любых проблем и штрафов.

**14. JS-фреймворки (библиотеки), преимущества и недостатки**

JavaScript-фреймворк — это инструмент, который вы можете использовать для разработки современных веб-приложений, особенно SPA.

В свое время веб-разработчики реализовали логику интерфейса, полагаясь в основном на vanilla JS и jQuery. Но по мере того, как приложения становились все более и более сложными, инструменты для решения этой проблемы стали развиваться.

**React**

React - это библиотека JavaScript, созданная разработчиками Facebook и Instagram. Согласно опросу Stack Overflow Survey 2017, React был признан самой популярной технологией среди разработчиков. React также имеет честь быть самым популярным проектом JavaScript, согласно количеству звезд на GitHub.

Так почему React получает все внимание? С помощью React можно создать интерактивный интерфейс с использованием декларативного подхода, в котором вы можете контролировать состояние приложения, говоря: «Представление должно выглядеть так». Он использует компонентную модель, в которой компоненты являются повторно используемыми элементами пользовательского интерфейса, и каждый компонент имеет свое собственное состояние. React использует виртуальную модель DOM, так что вам не нужно беспокоиться о прямом манипулировании с DOM. Другие примечательные особенности React включают однонаправленный поток данных, дополнительный синтаксис JSX и инструмент командной строки для создания проекта React с нуля.

**JQuery**

jQuery - это библиотека, которая сделала JavaScript более доступным а DOM-манипуляцией проще, чем раньше. Плавная кривая обучения jQuery и простой синтаксис породили на стороне клиента новое поколение новых разработчиков. Несколько лет назад jQuery считался прочным решением для создания надежных веб-сайтов с поддержкой кросс-браузерности. Основные функции jQuery, такие как манипулирование DOM на основе селекторов CSS, обработка событий и создание вызовов AJAX, подпитывали его популярность. Однако все изменилось, и среда JavaScript постоянно развивается. Некоторые функции jQuery были включены в новую спецификацию ECMAScript. Более того, новые библиотеки и фреймворки, используемые сегодня, имеют собственный способ связывания DOM, и поэтому простые методы манипуляции с DOM больше не требуются.

**D3: Документы, управляемые данными**

D3 (или D3.js) - мощная библиотека JavaScript для создания интерактивных визуализаций с использованием веб-стандартов, таких как SVG, HTML и CSS. В отличие от других библиотек визуализации, D3 предлагает лучший контроль над окончательным визуальным результатом.

D3 работает, связывая данные с DOM и затем преобразуя их в документ. Она также имеет собственную экосистему, которая состоит из плагинов и библиотек, которые расширяют ее базовые функциональные возможности. Библиотека существует с 2011 года, и в ней есть множество документации и учебников, которые помогут вам начать работу.

**Минусы**

* **Спецификация все еще развивается**

Спецификация веб-компонентов является и старой, и новой. Веб-компоненты были впервые представлены Алексом Расселом на конференции Fronteers 2011. Тем не менее, за последние год или два, их востребованность существенно возросла. Таким образом, в спецификации все еще много неразберихи. Например, импорт HTML является устаревшим, хотя большая часть документации / ресурсов по-прежнему ссылается на него.

* **Тестирование.** Существует не так много выделенных ресурсов для тестирования нативных веб-компонентов. Есть несколько многообещающих инструментов, таких как skatejs ssr и web component tester от Polymer. Но эти инструменты на самом деле предназначены для использования с соответствующими библиотеками. Это создает некоторые трудности для их использования с нативными веб-компонентами.
* **Обнаружение изменений.** Поддерживать синхронизацию состояния с представлением не так сложно в небольшом масштабе. Но все может очень быстро выйти из-под контроля, и вы обнаружите, что добавляете множество прослушивателей событий и селекторов запросов.

**Плюсы**

* **Размер пакета.** Конечный продукт может быть (с акцентом на может быть) намного меньше и компактнее, чем что-либо, разработанное с использованием современного фреймворка. Например, финальная сборка моего полнофункционального приложения с рецептами была меньше, чем половина сборки Angular.
* **Понимание.** Если вы на самом деле разрабатывали только с помощью фреймворка и его интерфейса командной строки, это может быть отличным упражнением для создания веб-приложения без дополнительных инструментов.
* **Простота и управление кодом.** Вы рискуете всякий раз, когда вводите сторонний код. Этот риск уменьшается с помощью проверенных и протестированных библиотек / фреймворков, но никогда не устраняется. Если вы пишите код самостоятельно или с вашей командой, вы можете уменьшить этот риск и поддерживать кодовую базу, которую вы знаете, как внутри, так и снаружи.

**15.** **Формат JSON**

JSON — это текстовый формат обмена данными, основанный на мультипарадигменном языке программирования. Его основное назначение состоит в хранении и передаче структурированного потока информации. При помощи простых правил формирования конструкций из символов в JavaScript, человек может обеспечить легкий и надежный способ хранения любого вида информации, будь то обычное число, целые строки или огромное количество различных объектов, выраженных в простом тексте.Помимо этого, формат JSON используется для объединения между собой объектов и структуры данных в виде набора компонентов, формируя тем самым программные единицы, позволяющие хранить и обрабатывать сложные записи, состоящие из нескольких переменных разного типа.После того как файл создан, содержащиеся в нем строки довольно легко перенаправить в другое положение Сети через любые пути передачи данных. Это связано с тем, что строка представляет собой обычный текст.

**Как устроен формат JSON**

В JSON типы данных подразделяются на несколько категорий: простые и сложные. К первому виду можно отнести, прежде всего, текстовые строки и числа, ко второму — объекты. В общей сложности выделяют шесть основных типов:

* Числительный. При этом числа могут быть как беззнаковыми целыми, так и целыми со знаком. В частности, здесь может содержаться дробная часть и представление действительных чисел в виде дробной части логарифма и порядка.
* Произвольная последовательность (строка) символов латинского алфавита, цифр и элементов пунктуации (от нуля и символов юникод).
* Литералы или константы, включаемые непосредственно в текст. Это может быть любое значение из true и false или их аналогов.
* Массив. Он представляет собой упорядоченный перечень символов от нуля и дальше.
* Объект. Это хаотично сложенный состав пар ключи/значение.
* Пустое значение, обозначающееся словом «Null».

**16. Программный интерфейс приложения API**

API — это интерфейс, позволяющий двум независимым компонентам программного обеспечения обмениваться информацией. Проще говоря, это то, что обеспечивает эффективный процесс коммуникаций между программами, использующими функции и ресурсы друг друга. API играет роль посредника между внутренними и внешними программными функциями, обеспечивая настолько эффективный обмен информацией, что конечные пользователи обычно его просто не замечают. Говоря по-простому, API действует как виртуальный посредник и передает информацию из одного интерфейса, например мобильного приложения, в другой. API связывает различные части программной платформы, чтобы передаваемая информация дошла до места назначения.Эти связующие узлы не только выполняют роль внутренних каналов связи, но и позволяют внешним инструментам получать доступ к этой же информации. Таким образом API-интерфейсы могут относиться к одной из двух категорий:

**Внутренние/частные** API(Частные API доступны только разработчикам и пользователям из числа сотрудников организации. Такие API обычно связывают внутренние процессы для уменьшения разрозненности рабочих данных и оптимизации совместной работы.)

**Внешние/открытые** API(Открытые API, в свою очередь, позволяют внешним разработчикам получать доступ к информации и интегрировать информацию, которая передается из одного программного инструмента в другой. Открытые или частные API экономят время разработчиков, позволяя им объединять платформы с имеющимися инструментами и устраняя необходимость в создании нового функционала с нуля.)

Вот несколько распространенных примеров использования API:

* Обмен информацией о рейсах между авиакомпаниями и туристическими сайтами
* Использование Google Maps в приложении для совместных поездок (райдшеринга)
* Создание виртуальных собеседников в службе обмена сообщениями
* Встраивание видеоклипов с YouTube на веб-странице
* Автоматизация рабочих процессов в программных инструментах для B2B-сектора

**17. Работа в редакторе Vim. Демонстрация**

**Режимы работы**

Основное отличие vim от многих других редакторов заключается в том, что в нем несколько режимов редактирования, а не один, как в большей части других. В других программах особые функции, например, копирование текста, приводится в действие комбинациями клавиш. В vim для разделения этих функций и набора текста используются разные режимы.

**Нормальный режим**

Режим по умолчанию, в котором vim находится после запуска. Применяется для быстрой навигации по документу и редактирования: в нем можно копировать, вставлять, перемещать, удалять и изменять текст. Переход из других режимов осуществляется нажатием клавиши «escape». Для набора текста не используется.

**Режим вставки**

Для набора текста нужно перейти в режим “вставки”. Он похож на интерфейс набора текста многих других программ. Вводимый вами текст появляется на экране в документе, каждая клавиша соответствует своему символу. Перейти в этот режим можно следующим образом:

i — в текущем положении курсора

a — на следующем символе после текущего положения

I — в начале текущей строки

A — в конце текущей строки

**Режим визуального выделения**

Позволяет визуально выделять текстовые фрагменты. Затем к ним можно применять команды нормального режима для редактирования или форматирования.

v — режим обычного (посимвольного) визуального выделения перемещением курсора вверх, вниз, влево или вправо.

V — режим построчного визуального выделения. Строки выделяются целиком перемещением вверх или вниз.

[ctrl]-v — режим блочного визуального выделения. Выделяется прямоугольный участок, который можно увеличивать и уменьшать. Позволяет выбирать фрагменты нескольких линий.

**Также есть дополнительные функции навигации, вот несколько наиболее важных:**

gg — переход в начало документа

G — переход в конец документа. Если предварительно ввести номер — переместиться на строку с этими номером

w — перемещение к следующему слову. Если предварительно ввести количество — переместиться на это количество слов

b — перемещение к предыдущему слову. Если предварительно вввести количество — переместиться на это количество слов назад

e — перемещение к концу слова. Если предварительно указать количество — переместиться на указанное количество слов

0 — перемещение к началу строки

$ — перемещение к концу строки

Что бы найти фрагмент текста, нужно перед текстом поиска ввести слэш «/» и нажать enter.

**Редактирование**

Редактирование выполняется посредством команд в нормальном режиме. Они очень эффективны в комбинации с командами перемещения. Любая команда из раздела перемещения может задать направление. Например, можно удалить или скопировать одно слово, указав “w” за соответствующей командой.

**Удаление текста**

x — удалить символ на позиции курсора

d — удалить в заданном командой перемещения направлении. Например, “dl” удаляет один символ справа

dd — удалить строку

D — удалить все символы от положения курсора до окончания строки

**Замена текста**

r — заменить символ на позиции курсора заданным после команды символом

c — заменить в заданном командой перемещения направлении. Например, “cb” заменяет предыдущее слово. Для набора текста на замену запускается режим вставки.

C — заменить до окончания строки. Текст на замену набирается в режиме вставки.

**Копирование и вставка**

y — копировать в заданном далее направлении

yy — копировать всю строку

Y — копировать до окончания строки

p — вставить последнюю скопированную (или удаленную!) строку ниже текущей

P — вставить последнюю скопированную (или удаленную!) строку выше текущей

**Дополнительные команды редактирования**

u — отменить последнее действие

<ctrl>-r — вернуть последнее отмененное действие

J — присоединить строку ниже к текущей

**Командный режим**

Используется для продвинутого редактирования, изменения параметров и управления. В нем выполняется сохранение, выход из программы, продвинутый поиск и многое другое. Активируется клавишей двоеточия “:”.

:q — выход из программы. Действие не будет выполнена, если не сохранить изменения

:q! — выход из программы с отменой всех изменений, которые не были сохранены

:w — сохранить изменения. При первичном сохранении или сохранении в другой файл укажите имя через пробел

:e — редактировать указанный далее файл

:bn — редактировать следующий файл (если открыто несколько файлов)

:bp — редактировать предыдущий файл (если открыто несколько файлов)

:qw – сохранить файл и выйти

**18. Базовые команды оболочки Bash/PowerShell. Примеры**

Оболочка, или шелл (shell) — это программа, в нашем случае названная «bash», что является сокращением от Bourne Again Shell. Оболочка принимает ваши команды и передаёт их операционной системе.

**pwd**

Команда pwd, сокращение от print working directory, отображает текущее местоположение в структуре каталогов.

**cd**

Команда cd позволяет перейти в новый каталог.

**mkdir**

Команда mkdir создаёт новый каталог в текущем каталоге.

**Основные команды**

**man**

Команда man отображает руководства по командам. Например, следующая команда выдаст всю информацию о команде cat

$ man cat

**cat**

Команда cat считывает файл, переданный как аргумент, и выводит его содержимое по стандартному каналу вывода. Передача нескольких файлов в виде аргумента приведёт к выводу конкатенированного содержимого всех файлов.

**echo**

Команда echo выводит свои аргументы по стандартному каналу вывода. Если вызвать echo без аргументов, будет выведена пустая строка.

$ echo Hello World

Hello World

**head**

Команда head читает первые 10 строк любого переданного текста и выводит их по стандартному каналу.

$ head -50 test.txt

**tail**

Команда tail работает аналогично команде head, но читает строки с конца:

$ tail -50 test.txt

Также можно просматривать добавляемые к файлу строки в режиме реального времени при помощи флага -f:

$ tail -f test.txt

**less**

Команда less позволяет перемещаться по переданному файлу или куску текста, причём в обоих направлениях.

$ less test.txt

$ ps aux | less

**true**

Команда true всегда возвращает ноль в качестве выходного статуса для индикации успеха.

**false**

Команда false всегда возвращает не-ноль в качестве выходного статуса для индикации неудачи.

**$?**

$? — это переменная, которая содержит выходной статус последней запущенной команды. Под статусом обычно понимается код возврата программы.

$ true

$ echo $?

0

$ false

$ echo $?

1

**grep**

Команда grep занимается поиском переданной строки в указанном файле:

$ cat users.txt

user:student password:123

user:teacher password:321

$ grep 'student` file1.txt

user:student password:123

**sed**

Команда sed — это потоковый редактор, преобразующий входные текстовые данные. Обычно её используют для замены выражений так: s/regexp/replacement/g. Например, следующий код заменит все слова «Hello» на «Hi»:

$ cat test.txt

Hello World

$ sed 's/Hello/Hi/g' test.txt

**ps**

Команда ps выводит информацию о запущенных процессах.

$ ps

PID TTY TIME CMD

35346 pts/2 00:00:00 bash Hi World

**export**

Команда export устанавливает переменные окружения для передачи дочерним процессам. Например, так можно передать переменную name со значением student:

$ export name=student

**Windows PowerShell** является мощным средством и может выполнять практически все, что хочет человек на своем компьютере. Но единственная проблема заключается в том, что это инструмент командной строки и не имеет графического интерфейса.

* PowerShell — отличный инструмент, который можно использовать для запуска приложений UWP за считанные секунды. Ты можешь использовать

Start-Process "ms-settings:"

* Получите справку о любом командлете

Get-Help

Получить помощь

Get-Help -Full

Get-Help -Example

Get-Help \*

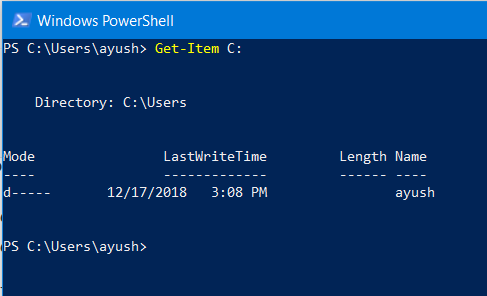
Здесь первая запись расскажет вам, как использовать этот командлет. Вторая запись даст вам простое резюме конкретного командлета. Третья запись даст подробную информацию о соответствующем командлете. Четвертая запись будет содержать все, что показывает третий командлет, но добавит пример того, как использовать этот командлет. И, наконец, пятый командлет перечислит каждую команду, которая доступна для вашего использования.

* Чтобы найти команды аналогичного типа или содержащие в себе определенную фразу, вы можете использовать командлет Get-Command

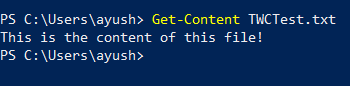
Get-Command -Name

Get-Command –CommandType

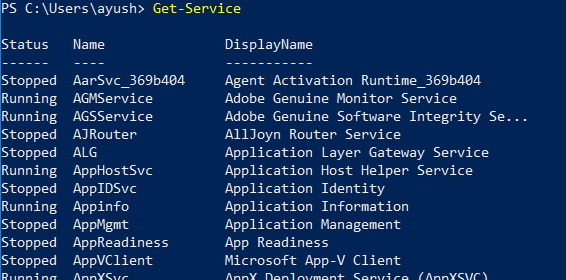
* Поиск определенного файла



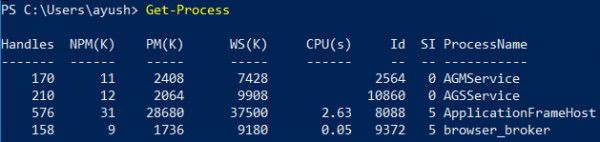
* Прочитайте содержимое файла



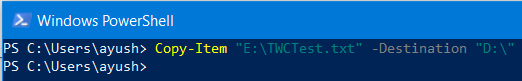
* Прочтите информацию обо всех службах на компьютере



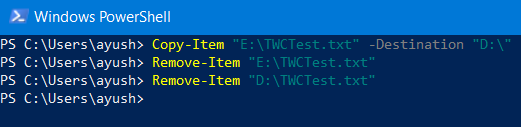
* Читайте информацию обо всех процессах на компьютере



* Скопируйте файл или каталог



* Удалить файл или каталог



**19. Принцип работы системы контроля версий Git**

Система контроля версий — это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии. Для контроля версий файлов в этой книге в качестве примера будет использоваться исходный код программного обеспечения, хотя на самом деле вы можете использовать контроль версий практически для любых типов файлов.

VCS отслеживает изменения в файлах, предоставляет возможности для создания новых и слияние существующих ветвей проекта, производит контроль доступа пользователей к проекту, позволяет откатывать исправления и определять кто, когда и какие изменения вносил в проект. Основным понятием VCSявляется репозиторий (repository) – специальное хранилище файлов и папок проекта, изменения в которых отслеживаются. В распоряжении разработчика имеется так называемая “рабочая копия” (working copy) проекта, с которой он непосредственно работает. Рабочую копию необходимо периодически синхронизировать с репозиторием, эта операция предполагает отправку в него изменений, которые пользователь внес в свою рабочую копию (такая операция называется commit) и актуализацию рабочей копии, в процессе которой к пользователю загружается последняя версия из репозитория (этот процесс носит название update)

**20. Основные команды Git. Примеры**

• **git config**

Она может быть использована для указания пользовательских настроек, таких как электронная почта, имя пользователя, формат и т.д. К примеру, данная команда используется для установки адреса электронной почты:

git config --global user.email адрес@gmail.com

**• git init**

Эта команда используется для создания GIT репозитория. Пример использования:

git init

**• git add**

Команда git add может быть использована для добавления файлов в индекс.

git add temp.txt

**• git clone**

Команда git clone используется для клонирования репозитория

git clone имя.пользователя@хост:/путь/до/репозитория

И наоборот, для клонирования локального репозитория используйте:

git clone /путь/до/репозитория

**• git commit**

Команда git commit используется для коммита изменений в файлах проекта.

git commit –m “Сообщение идущее вместе с коммитом”

**• git status**

Команда git status отображает список измененных файлов, вместе с файлами, которые еще не были добавлены в индекс или ожидают коммита. Применение:

git status

**git branch**

Команда git branch может быть использована для отображения, создания или удаления веток. Для отображения всех существующих веток в репозитории введите:

git branch

Для удаления ветки:

git branch –d <имя-ветки>

**• git pull**

Команда pull используется для объединения изменений, присутствующих в удаленном репозитории, в локальный рабочий каталог. Применение:

git pull

**• git merge**

Команда git merge используется для объединения ветки в активную ветвь. Применение:

git merge <имя-ветки>