МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
Факультет информационных технологий

УТВЕРЖДЕНО
Декан факультета
Информационных технологий
_____ / Д.Г. Демидов /
«16» ____02_____ 2023 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

«Веб-разработка на стороне клиента»

Направление подготовки/специальность
09.03.01 Информатика и вычислительная техника

Профиль/специализация
«Веб-технологии»

Квалификация
бакалавр

Формы обучения
очная

Москва, 2023 г.

Разработчик(и):

к.т.н., доцент / В.Ю. Верещагин /

Согласовано:
Заведующий кафедрой «Инфокогнитивные технологии»,

к.т.н., доцент / Е.А. Пухова /

Содержание

1 Цели, задачи и планируемые результаты обучения по дисциплине

К **основным целям** освоения дисциплины «Веб-разработка на стороне клиента» относится:

- изучение языков программирования java script и type script;
- получение знание и умений разработки single page application с помощью фреймворка angular;
- овладение общей методикой разработки веб-приложений;
- закрепление получаемых в семестре знаний и навыков на практике;
- формирование взаимосвязей, получаемых в семестре знаний и навыков с изученными ранее и изучаемых параллельно с данной дисциплиной;
- подготовка студентов к деятельности в соответствии с квалификационной характеристикой бакалавра.

К **основным задачам** дисциплины относятся:

- овладение навыками и приемами программирования frontend;
- изучение и освоение теоретического материала, как в процессе контактной, так и в ходе самостоятельной работы;
- выполнение предоставленных практических заданий различных форм, как в процессе контактной, так и в ходе самостоятельной работы;
- самостоятельная работа над тематикой дисциплины для формирования компетенций основной образовательной программы (далее, ООП).

Обучение по дисциплине «Веб-разработка на стороне клиента» направлено на формирование у обучающихся следующих компетенций:

Код и наименование компетенций	**Индикаторы достижения компетенции**
ПК- 3. Способен разрабатывать требования и проектировать программное обеспечение	ПК-3.1. Знать: методы и средства проектирования программных интерфейсов. ПК-3.2. Уметь: применять методы и средства проектирования программного обеспечения, структур данных, баз данных, программных интерфейсов. ПК-3.3. Владеть: современным инструментарием и средами разработки программного кода;

	современным инструментарием и средами проектирования программного кода.
ПК-5. Способен проводить работы по интеграции программных модулей и компонент и проверку работоспособности выпусков программных продуктов	ПК-5.1. Знать: основные способы верстки web-страниц, современные языки разметки; ПК-5.3. Владеть: навыками использования языка разметки гипертекста с языками программирования; навыками работы с веб-технологиями и программирования;

2 Место дисциплины в структуре образовательной программы

Дисциплина «Веб-разработка на стороне клиента» относится к числу учебных обязательных дисциплин основной профессиональной образовательной программы.

Дисциплина взаимосвязана логически и содержательно-методически со следующими дисциплинами и практиками ОПОП:

- Серверная веб-разработка;
- Алгоритмическое программирование;
- Основы Веб-технологий;
- Индексирование текстов и информационный поиск;
- Разработка мобильных приложений;
- Основы веб-разработки на стороне клиента

3 Структура и содержание дисциплины

Общая трудоемкость дисциплины составляет 5 зачетных единицы, т.е. 180 академических часов (из них 118 часов – самостоятельная работа студентов и 62 часа – аудиторные занятия).

Разделы дисциплины изучаются на втором курсе в третьем семестре, форма промежуточной аттестации – экзамен, курсовой проект.

3.1 Виды учебной работы и трудоемкость для очной формы обучения

№ п/п	Вид учебной работы	Количество часов	Семестры	
			3	
1	**Аудиторные занятия**	**62**	62	
	В том числе:			
1.1	Лекции	18	18	
1.2	Семинарские/практические занятия			
1.3	Лабораторные занятия	44	44	
2	**Самостоятельная работа**	**118**	118	
	Курсовой проект			

3	**Промежуточная аттестация**			
	Экзамен		экзамен	
	Итого:	**180**	180	

3.2 Тематический план изучения дисциплины для очной формы обучения

№ п/п	Разделы/темы дисциплины	Трудоемкость, час					
		Всего	Аудиторная работа				Самостоятельная работа
			Лекции	Семинарские/ практические занятия	Лабораторные занятия	Практическая подготовка	
1	JavaScript. Объекты и классы	16	2		4		10
2	JavaScript. Модули	16	2		4		10
3	JavaScript. Массивы и коллекции	15	1		4		10
4	Webpack, Eslint, babel	16	2		4		10
5	TypeScript. Примитивные типы данных (системы типов)	16	2		4		10
6	TypeScript. Ссылочные типы	13	1		2		10
7	TypeScript. Классы, модули и интерфейсы	15	1		4		10
8	TypeScript. Литералы и дженерики	15	1		4		10
9	Vue. Vue CLI	16	2		4		10
10	Vue. Маршрутизация	16	2		4		10
11	Vue Router	11	1		2		8
12	Vuex	15	1		4		10
13	Итоговая аттестация						
Итого		**180**	**18**		**44**		**118**

3.3 Содержание дисциплины

JavaScript. Объекты и классы

Тема раскрывает основы

Основы класса: создание, наследование, переопределение методов, использование конструкторов, статических методов и свойств.

Методы и свойства класса: определение и использование методов класса, объявление и использование свойств класса.

Наследование: создание подклассов, переопределение и переопределение свойств и методов родительского класса.
Создание и использование объектов: создание объектов класса, использование методов и свойств объекта.
Полиморфизм: использование наследования и переопределения для создания полиморфных объектов.

JavaScript. Модули
Тема раскрывает механизмы деления программ и запросов на JavaScript на отдельные файлы (модули), что позволяет составлять сложные программы.

JavaScript. Массивы и коллекции
В данной теме обучающиеся знакомятся с массивами и строками, а также учатся применять к ним запросы.

Webpack, Eslint, babel
Данная тема посвящена основным компонентам JavaScript в частности с такими, как webpack – сборщиком модулей с открытым исходным кодом, eslint – инструментом статического анализа кода для выявления проблемных шаблонов, а также с babel – бесплатным транскомпилятором с открытым исходным кодом.

TypeScript. Примитивные типы данных (системы типов)
Изучение данной темы позволяет обучающемуся расширить базовые возможности JavaScript с целью разработки веб-приложений при помощи инструмента TypeScript, языка программирования, представленного компанией Microsoft в 2012 году. Как и любой другой язык программирования, изучение TypeScript начинается с базовых типов данных (Value type).

TypeScript. Ссылочные типы
После изучения базовых типов, хранящих сами значения, обучающийся изучает ссылочные типы (Reference types), которые хранят ссылку на значение.

TypeScript. Классы, модули и интерфейсы
Тема посвящена объектно-ориентированному подходу при написании и кода на TypeScript, созданию пользовательских типов данных с возможностью встраивания в них методов (функций).

TypeScript. Литералы и дженерики
Тема знакомит обучающегося с литералами, что дает возможность использования строк и чисел в качестве типов данных, а также с дженерик-типами, использозвание которых позволяет создавать компоненты, которые будут работать с несколькими типами, а не с каким-то определенным, что помогает сделать компонент более переиспользуемым.

Vue. Vue CLI
Поскольку на практике при разработке приложений использование чистого кода JavaScript упирается в недостаток возможностей и гибкости используется Vue CLI, который позволяет использовать более надежные средства для улучшения процесса разработки.

Vue. Маршрутизация
В данной теме обучающийся изучает Vue.js, который имеет полноценную систему маршрутизации, что позволяет сопоставлять запросы к приложению с определенными компонентами. За работу системы маршрутизации во Vue.js отвечает специальная библиотека - vue-router.

Vue Router
В данной теме изучается Vue Router – официальная библиотека маршрутизации для Vue.js, которая позволяет оптимизировать создание SPA-приложений, что повышает быстродействие и скорость загрузки системы.

Vuex
Vuex – это библиотека управления состоянием, созданная командой Vue для управления данными в приложениях Vue.js.

3.4 Тематика лабораторных занятий

1. **Лабораторная работа №1.** Простейшая программа на JS. Hello world. Простые типы данных. Операторы условия. Циклические алгоритмы.
2. **Лабораторная работа №2.** String. Методы для работы со строками. Array. Методы для работы с массивами
3. **Лабораторная работа №3.** Регулярные выражения.
4. **Лабораторная работа №4.** Функции в JS.
5. **Лабораторная работа №5.** Библиотека Math
6. **Лабораторная работа №6.** Объекты в JS. ES6 классы в JS
7. **Лабораторная работа №7.** События.
8. **Лабораторная работа №8.** Изучение приемов для работы с асинхронным кодом в JS
9. **Лабораторная работа №9.** Язык typescript. Фреймворк angular.
10. **Лабораторная работа №10.** Angular components
11. **Лабораторная работа №11.** «Обмен данными между компонентами и формы в angular»
12. **Лабораторная работа №12.** Пайпы в angular
13. **Лабораторная работа №13.** Сервисы в angular
14. **Лабораторная работа №14.** Модули в angular
15. **Лабораторная работа №15.** Роутинг в angular

3.5 Тематика курсовых проектов

1. **Управление задачами и Kanban-доска (Vue.js, Vuex).**

Создание приложения для управления задачами с интерфейсом доски Канбан.

Используйте Vue.js и Vuex для управления состояниями, чтобы организовать задачи по столбцам (например, To-Do, In Progress, Done).

Основные возможности:

- Добавление, редактирование и удаление задач с поддержкой перетаскивания между столбцами.
- Аутентификация пользователей и доски задач для конкретных пользователей.
- Совместная работа с досками задач с обновлением в реальном времени.
- Даты выполнения, метки и описания задач.
- Фильтрация и поиск задач.

2. **Интерактивная панель визуализации данных (Vue.js, D3.js).**

Разработка панели визуализации данных, отображающей динамические графики и диаграммы.

Используйте Vue.js для фронтенда и D3.js для создания интерактивных визуализаций данных.

Ключевые особенности:

- Подключение к публичному API или использование имитационных данных для визуализации.
- Создание различных типов диаграмм, таких как гистограммы, линейные диаграммы, круговые диаграммы и т.д.
- Реализовать удобные для пользователя функции взаимодействия, такие как всплывающие подсказки, фильтрация и сортировка.
- Предоставить пользователям возможность настраивать данные, отображаемые на приборной панели.

3. **Трекер личных финансов (Vue.js, Firebase, Chart.js)**

Создайте трекер личных финансов, который поможет пользователям управлять своими доходами, расходами и бюджетом.

Используйте Vue.js для фронтенда, Firebase для хранения данных и Chart.js для визуализации финансовых данных.

Ключевые особенности:

- Аутентификация пользователей и персонализированные финансовые профили.
- Учет доходов и расходов с помощью категорий и тегов.
- Визуализация финансовых данных с помощью интерактивных графиков и диаграмм.
- Установка ежемесячных бюджетов и получение уведомлений о приближении к лимитам.
- Формирование отчетов и сводок о финансовой деятельности.

4. **Интерактивное приложение для создания историй (Vue.js, CSS-анимация):**

Создайте интерактивное приложение для чтения историй, в котором пользователи могут читать или участвовать в историях в стиле "выбери сам".

Используйте Vue.js для фронтенда, чтобы управлять разветвлением сюжета и выбором.

Ключевые особенности:

- Несколько историй с разветвленным повествованием.
- Динамичные и увлекательные анимации или переходы между сюжетными сценами.
- Предоставление пользователям возможности выбора, влияющего на исход истории.
- Сохранение и загрузка прогресса, чтобы пользователи могли продолжить работу с того места, на котором остановились.
- Простой инструмент создания историй для авторов.

5. **Платформа для проведения онлайн-викторин (Vue.js, Firebase):**

Создайте платформу для онлайн-викторин, где пользователи могут создавать, делиться и проходить тесты на различные темы.

Используйте Vue.js для фронтенда и Firebase для управления пользователями и хранения тестов.

Ключевые особенности:

- Регистрация и аутентификация пользователей.
- Создание тестов с различными типами вопросов (множественный выбор, верно/неверно и т.д.).
- Совместный доступ к тестам и возможность публичного/приватного доступа к ним.
- Таблицы лидеров для отслеживания результатов пользователей.
- Удобное прохождение теста.

6. **Таймер Pomodoro (Vue.js, localStorage):**

Создайте приложение с таймером Pomodoro, которое поможет пользователям управлять своей работой с помощью временных интервалов.

Используйте Vue.js для фронтенда и localStorage для хранения данных.

Основные возможности:

- Установка пользовательских интервалов Pomodoro (время работы и перерыва).
- Визуальный таймер обратного отсчета с индикатором выполнения.
- Звуковые уведомления о периодах работы и перерыва.
- Интеграция со списком задач для сосредоточенной работы.
- Отслеживание статистики и производительности.

4 Учебно-методическое и информационное обеспечение

4.1 Нормативные документы и ГОСТы

1. Федеральный закон от 29 декабря 2012 года № 273-ФЗ «Об образовании в Российской Федерации» (с изменениями и дополнениями);

2. Федеральный государственный образовательный стандарт высшего образования - бакалавриат по направлению подготовки 09.03.01 Информатика и вычислительная техника, утвержденный Приказом Министерства образования и науки РФ от 19 сентября 2017 г. N 929 "Об утверждении федерального… Редакция с изменениями N 1456 от 26.11.2020

3. Приказ Министерства образования и науки РФ от 05 апреля 2017 г. № 301 «Об утверждении Порядка организации и осуществления образовательной деятельности по образовательным программам высшего образования – программам бакалавриата, программам специалитета, программам магистратуры;

4. Порядок проведения государственной итоговой аттестации по образовательным программам высшего образования – программам бакалавриата, программам специалитета и программам магистратуры, утвержденный приказом Минобрнауки России от 29 июня 2015 г. № 636;

5. Положение о практической подготовке обучающихся, утвержденное приказом Министерства науки и высшего образования Российской Федерации и Министерства просвещения Российской Федерации от 5 августа 2020 г. № 885/390;

6. Устав и локальные нормативные акты Московского политеха

Области профессиональной деятельности и сферы профессиональной деятельности, в которых выпускники, освоившие программу бакалавриата (далее - выпускники), могут осуществлять профессиональную деятельность:

06 Связь, информационные и коммуникационные технологии (в сфере проектирования, разработки, внедрения и эксплуатации средств вычислительной техники и информационных систем, управления их жизненным циклом)

Выпускники могут осуществлять профессиональную деятельность в других областях и (или) сферах профессиональной деятельности при условии соответствия уровня их образования и полученных компетенций требованиям к квалификации работника, предъявляемым соответствующими профессиональными стандартами.

4.2 Основная литература

Полуэктова, Н. Р. Разработка веб-приложений : учебное пособие для вузов / Н. Р. Полуэктова. — Москва : Издательство Юрайт, 2023. — 204 с. — (Высшее образование). — ISBN 978-5-534-13715-6. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/519714

4.3 Дополнительная литература

Верещагин, В.Ю. Создание веб-страниц на стороне клиента: учебно- методическое пособие для студентов бакалавриата, обучающихся по направлению подготовки 09.03.01 «Информатика и вычислительная техника», профиль «Веб-технологии» / В.Ю. Верещагин, М.В. Даньшина. – Москва: Московский Политех, 2023. – 1 CD-R. – Загл. с титул. экрана. – Текст: электронный. ISBN 978-5-2760-2786-9. — URL: https://online.mospolytech.ru/mod/data/view.php?d=127&rid=5281&filter=1

4.4 Электронные образовательные ресурсы

1. Курс ЭОР в lms Веб разработка на стороне клиента
https://online.mospolytech.ru/course/view.php?id=12775

4.5 Лицензионное и свободно распространяемое программное обеспечение

1. Visual Studio Code
2. Браузеры Chrome, Edge, Firefox
3. OpenVPN с правами для запуска у студентов
4. FileZilla
5. PuTTy
6. Git
7. Node.js 18
8. Python 3.10
9. Wireshark

4.6 Современные профессиональные базы данных и информационные справочные системы

1. https://doka.guide/
2. https://developer.mozilla.org/ru/
3. https://roadmap.sh/frontend

5 Материально-техническое обеспечение

Для проведения лабораторных работ и самостоятельной работы студентов подходят аудитории, оснащенные компьютерами с программным обеспечением в соответствии со списком в пункте 4.5 и подключенные к интернету.

Число рабочих мест в аудитории должно быть достаточным для обеспечения индивидуальной работы студентов.

Рабочее место преподавателя должно быть оснащено компьютером с подключенным к нему проектором или иным аналогичным по функциональному назначению оборудованием.

6 Методические рекомендации

6.1 Методические рекомендации для преподавателя по организации обучения

1. При подготовке к занятиям следует предварительно проработать материал занятия, предусмотрев его подачу точно в отведенное для этого время занятия. Следует подготовить необходимые материалы – теоретические сведения, задачи и др. При проведении занятия следует контролировать подачу материала и решение заданий с учетом учебного времени, отведенного для занятия.

2. При проверке работ и отчетов следует учитывать не только правильность выполнения заданий, но и оптимальность выбранных методов решения, правильность выполнения всех его шагов.

6.2 Методические рекомендации для обучающихся по освоению дисциплины

Изучение дисциплины осуществляется в строгом соответствии с целевой установкой в тесной взаимосвязи учебным планом. Основой теоретической подготовки студентов являются лекции и самостоятельная работа.

В процессе самостоятельной работы студенты закрепляют и углубляют знания, полученные во время аудиторных занятий, готовятся к промежуточной аттестации, а также самостоятельно изучают отдельные темы учебной программы.

На занятиях студентов, в том числе предполагающих практическую деятельность, осуществляется закрепление полученных, в том числе и в процессе самостоятельной работы, знаний. Особое внимание обращается на развитие умений и навыков установления связи положений теории с профессиональной деятельностью будущего специалиста.

Самостоятельная работа осуществляется индивидуально. Контроль самостоятельной работы организуется в двух формах:

- самоконтроль и самооценка студента;
- контроль со стороны преподавателей (текущий и промежуточный).

Текущий контроль осуществляется на аудиторных занятиях.

Критериями оценки результатов самостоятельной работы студента являются:

- уровень освоения студентом учебного материала;
- умения студента использовать теоретические знания при выполнении практических задач;
- сформированность компетенций;
- оформление материала в соответствии с требованиями.

Приветствуется обсуждение самих заданий с другими студентами: можно как давать, так и получать советы по общей стратегии выполнения и изучения материала, давать и получать помощь в отладке. Однако писать код студент должен самостоятельно. Делиться кодом или писать его совместно запрещено.

7 Фонд оценочных средств

7.1 Методы контроля и оценивания результатов обучения

Приведенные ниже правила выставления оценок и опозданий могут быть изменены, если преподаватель сочтет это необходимым. Важно, чтобы студенты регулярно просматривали план курса, выложенный в СДО, на предмет его обновления или изменения.

Достижение компетенций оценивается с помощью лабораторных работ и рубежных контролей.

В соответствии с планом на дисциплины студентам выдаются задания на лабораторные работы. Помимо требований и описания функционала в работе указан крайний срок сдачи. Для сдачи лабораторной работы студенту необходимо прислать ссылку на репозиторий в GitHub Classroom и на хостинг, где размещен результат работы с реализованным функционалом, описанным в задании. Работа считается сданной если в ней реализовано 80% и более требований и функционала, описанного в задании.

Каждый студент имеет право на 6 дней опоздания, которые могут быть потрачены на любые задания в течение семестра. Опоздания предназначены для решения особых ситуаций, таких как болезнь или чрезвычайные семейные обстоятельства.

Когда использованы все дни опоздания за каждый день просрочки начисляется штраф в размере 25% от максимального результата за задание. Задания, присланные позже, чем 4 дня, не будут оцениваться. В связи с зависимостью между работами студентам может потребоваться все равно выполнить предыдущие работы, даже если они не оцениваются.

После сдачи лабораторной работы студент должен ее защитить. Во время защиты лабораторной работы преподаватель проверяет репозиторий, хостинг и выполнение критериев и требований задания, а студент отвечает на вопросы преподавателя по его коду, а также теоретических вопросов, приведенных после текста задания лабораторной работы. Если студент отказывается отвечать на вопросы, или дает полностью неверные ответы, или ответы не по теме, то работа может считаться сданной, но при этом она не оценивается.

Работа должна быть выполнена студентом самостоятельно: в репозитории в системе контроля версий студента содержатся коммиты только за его авторством, по этим коммитам можно проследить как велась работа, студент может объяснить свой код и ход выполнения работы, если эти правила не соблюдаются, то работа не считается сданной и не оценивается.

Студенты должны заранее сообщать о том, что у них могут возникнуть трудности со своевременной сдачей задания или проекта. При наличии реальных причин задержки студентам следует как можно скорее связаться с преподавателем и обсудить возможные условия.

7.2 Шкала и критерии оценивания результатов обучения

Лабораторная работа оценивается в процентах степени выполнения следующих критериев и для выставления оценки суммируются проценты за каждый из четырех критериев:

1. Полнота выполнения практического задания (30%): соответствует ли функциональность заданным требованиям и целям, насколько точно и без ошибок код выполняет поставленные задачи, насколько эффективно задание отвечает требованиям целевой аудитории и обеспечивает приятное восприятие.
2. Качество и структура кода (10%): качество, читаемость и организация кода, рациональность выполнения задания, последовательность именования и соблюдение лучших практик.
3. Творчество и инновации (10%): творческий подход студентов к выполнению заданий, насколько студенты вышли за рамки основных требований и реализовали дополнительные возможности или использовали уникальные решения.
4. Ответы на вопросы по коду студента и теории (50%):

Дает краткий ответ, содержащий ошибки или неточности. На наводящие вопросы отвечает неправильно (10% из 50%)

Дает развернутый ответ, содержащий ошибки или неточности. На наводящие вопросы отвечает неверно (20% из 50%)

Дает развернутый ответ, содержащий ошибки или неточности. На наводящие вопросы отвечает правильно (30% из 50%)

Дает правильные и развернутые ответы на вопросы (50% из 50%).

R лабораторные рассчитывается как среднее результатов за все лабораторные работы. За полное и безошибочное выполнение всех лабораторных работ в срок и их защиту можно получить максимум 100 баллов (R лабораторные).

Также имеется коэффициент сданных работ K сданные, который равен 1 если все работы сданы и 0 если хотя бы одна работа не сдана.

Итоговый балл рассчитывается по формуле: R сем = R лабораторные * K сданные.

Итоговый балл пересчитывается по шкале ниже и на основании полученной оценки фиксируется результат промежуточной аттестации.

Соответствие баллов в 100 балльной рейтинговой системе оценке по 4-бальной шкале:
0-54 - неудовлетворительно
55-69 - удовлетворительно
70-84 - хорошо
85-100 – отлично

7.3 Оценочные средства

7.3.1 Промежуточная аттестация

Примерный список вопросов к экзамену:

1. Что такое JS?
2. Какой основное назначение JS?
3. В чем отличие статического и динамического контента?
4. Как внедрить JS-сценарий на статическую страницу?
5. Какие общепринятые требования к страницам с JS кодам?
6. Где выполняется JS-код?
7. Как вывести данные на страницу, используя JS?
8. Как с помощью JS получить данные от пользователя?
9. Как в JS-программе указать условие выполнения части кода?
10. Что такое переменная?
11. Что такое цикл?
12. Что такое условный оператор?
13. Какие виды циклов бывают?
14. Чем цикл с предусловием отличается от цикла с постусловием?
15. Чем отличаются операторы break и continue?
16. Что такое итерация?
17. Какие операторы доступны в JS?
18. Что такое строка?
19. Что такое массив?
20. Что такое цикл for in?
21. Что такое цикл for of?
22. Описание метода toString()
23. Описание метода join()

24. Описание метода reverse()
25. Описание метода sort()
26. Описание метода concat()
27. Описание метода slice()
28. Описание метода splice()
29. Описание метода push()
30. Описание метода pop()
31. Описание метода unshift()
32. Описание метода shift()
33. Описание метода indexOf()
34. Описание метода split()
35. Описание метода replace()
36. Описание метода toLowerCase()
37. Описание метода toUpperCase()
38. Описание метода match()
39. Что такое регулярные выражения?
40. Как использовать регулярные выражения в js сценарии?
41. Что такое пользовательская функция?
42. Как функция возвращает значение?
43. Можно ли вызвать из функции другую функцию?
44. Можно ли вызвать из функции эту же функцию?
45. Продолжит ли функция свою работу после выполнения инструкции return?
46. Сколько раз инструкция return может быть использована в теле функции?
47. Сколько аргументов может быть передано функции?
48. Что-такое аргументы "по умолчанию"?
49. Может ли функция вообще не иметь аргументов?
50. Должны ли совпадать имена переменных-аргументов при объявлении и при вызове функции?
51. В каких случаях имеет смысл использовать пользовательские функции?
52. Может ли функция принимать в параметрах другую функцию?
53. Что такое контекст вызова?
54. Отличается ли контекст вызова у стрелочных функций?
55. Как использовать функции обратного вызова?
56. Как сгенерировать случайное число в заданном диапазоне?
57. Как округлить число?
58. Что такое объект?
59. Что такое литералы объекта?
60. Что такое классы?
61. Что такое свойства классов?
62. Что такое методы классов?
63. Что такое статические свойства классов?
64. Что такое статические методы классов?
65. Что такое наследование?
66. Что такое события?
67. Какие основные события мыши?
68. Какие основные события клавиатуры?

69. Какие основные события формы?
70. Почему организовывать асинхронный код функциями обратного вызова – это чаще всего плохая идея?
71. Что такое обещания?
72. Описание метода then?
73. Обработка исключений
74. Как использовать await?
75. Что такое typescript?
76. Для чего нужны типизация данных?
77. Синтаксис typescript
78. Команда для создания нового приложения на angular
79. Что такое angular?
80. Для чего нужны компоненты?
81. Что такое шаблон в компоненте
82. Использование циклов и оператора условия в шаблоне
83. Способы передачи данных внутри компонента
84. Как получить данные внутри компонента из вне?
85. Что такое eventEmmitor?
86. Какой класс служить для создания форм?
87. Как происходит валидация форм?
88. Как получить данные внутри компонента из вне?
89. Что такое eventEmmitor?
90. Какой класс служить для создания форм?
91. Как происходит валидация форм?
92. Что такое сервис?
93. Какими бывают http запросы?
94. Какой класс в angular используется для создания http запросов?
95. Что такое модули?
96. Зачем разбивать приложение на модули?
97. Что такое guard?
98. Что значит одностраничное приложение?
99. Какие преимущества у одностраничных приложений?
100. Как организуется роутинг в приложениях на angular?