

Документ подписан простой электронной подписью

Информация о владельце: МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФИО: Максимов Алексей Борисович
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

Должность: директор департамента по образовательной политике

Дата подписания: 30.09.2026 22:54:19

Уникальный программный ключ:

8db180d1a3f02ac9e60521a5672742735c18b1d6

**«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(МОСКОВСКИЙ ПОЛИТЕХ)**

Факультет Информационные технологии

УТВЕРЖДАЮ

Декан факультета

«Информационных технологий»


_____/ Д.Г. Демидов /
« 26 » _____ февраля 2026 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

«Низкоуровневые веб-технологии»

Направление подготовки/специальность

09.03.01 Информатика и вычислительная техника

Профиль/специализация

«Веб-технологии и продуктовый дизайн»

Квалификация

бакалавр

Формы обучения

очная, заочная

Год начала обучения – 2026 г.

Москва, 2026 г.

Разработчик(и):

Разработчик(и):

К.Т.Н., доцент

 / Е.А. Пухова /

К.Т.Н.

 /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	4
2	Место дисциплины в структуре образовательной программы	4
3	Структура и содержание дисциплины	4
3.1	Виды учебной работы и трудоемкость для очной формы обучения	5
3.2	Тематический план изучения дисциплины для очной формы обучения	5
3.3	Содержание дисциплины	7
3.4	Тематика лабораторных занятий	6
4	Учебно-методическое и информационное обеспечение	8
4.1	Нормативные документы и ГОСТы	8
4.2	Основная литература	8
4.3	Дополнительная литература	9
4.4	Электронные образовательные ресурсы	9
4.5	Лицензионное и свободно распространяемое программное обеспечение	9
4.6	Современные профессиональные базы данных и информационные справочные системы	9
5	Материально-техническое обеспечение	9
6	Методические рекомендации	10
6.1	Методические рекомендации для преподавателя по организации обучения	10
6.2	Методические рекомендации для обучающихся по освоению дисциплины	10
7	Фонд оценочных средств	11
7.1	Методы контроля и оценивания результатов обучения	11
7.2	Шкала и критерии оценивания результатов обучения	11
7.3	Оценочные средства	12

1 Цели, задачи и планируемые результаты обучения по дисциплине

К основным целям освоения дисциплины «Низкоуровневые веб-технологии» относится:

- получение знаний и умений углубленно работать с памятью и Foreign Function Interface на уровне браузера;
- овладение общей методикой применения потоков в браузере;
- закрепление получаемых в семестре знаний и навыков на практике;

К основным задачам освоения дисциплины «Низкоуровневые веб-технологии» следует отнести:

- ознакомление с областями применения низкоуровневых веб-технологий,
- овладение навыками и приемами работы с Web Components, Web Workers, WebAssembly;
- 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;

Обучение по дисциплине «Низкоуровневые веб-технологии» направлено на формирование у обучающихся следующих компетенций:

Код и наименование компетенций	Индикаторы достижения компетенции
ПК-5. Способен осуществлять концептуальное, функциональное и логическое проектирование систем среднего и крупного масштаба и сложности	ИПК5.2. Умеет: формулировать цели, исходя из анализа проблем, потребностей и возможностей; использовать основные приемы web-дизайна. Внедрять графические, Звуковые, анимационные объекты в страницу; ИПК-5.3. Владеет: навыками концептуального, функционального и логического проектирования; средствами работы со средствами Internet и Web-технологий для решения задач профессиональной деятельности;
ПК-4. Способен проводить работы по интеграции программных модулей и компонент, и проверку работоспособности выпусков программных продуктов	ИПК-4.1. Знает: основные способы верстки web-страниц, современные языки разметки; ИПК-4.3. Владеет: навыками использования языка разметки гипертекста с языками программирования; навыками работы с веб-технологиями и программирования.

2 Место дисциплины в структуре образовательной программы

Дисциплина относится к части, формируемой участниками образовательных отношений блока Б1.2 и междисциплинарно связана с дисциплинами:

- Основы веб-технологий;
- Основы веб-разработки на стороне клиента;
- Проектирование пользовательских интерфейсов в веб

3 Структура и содержание дисциплины

Общая трудоемкость дисциплины составляет 2 зачетные единицы (72 часа).

3.1 Виды учебной работы и трудоемкость для очной формы обучения

3.1.1 Очная форма обучения

№ п/п	Вид учебной работы	Количество часов	Семестры	
			5	
1	Аудиторные занятия	36	36	
	В том числе:			
1.1	Лекции	18	18	
1.2	Семинарские/практические занятия	-	-	
1.3	Лабораторные занятия	18	18	
2	Самостоятельная работа	36	36	
3	Промежуточная аттестация			
	Зачет/диф.зачет/экзамен		зачет	
	Итого:	72	72	

3.1.2 Заочная форма обучения

№ п/п	Вид учебной работы	Количество часов	Семестры	
			6	
1	Аудиторные занятия	8	8	
	В том числе:			
1.1	Лекции	4	4	
1.2	Семинарские/практические занятия			
1.3	Лабораторные занятия	4	4	
2	Самостоятельная работа	64	64	
3	Промежуточная аттестация			
	Зачет/диф.зачет/экзамен	зачет	зачет	
	Итого:	72	72	

3.2 Тематический план изучения дисциплины для очной формы обучения

3.2.1 Очная форма обучения

№ п/ п	Разделы/темы дисциплины	Трудоемкость, час					
		Всего	Аудиторная работа				Самостоятельная работа
			Лекции	Семинарские/ практические занятия	Лабораторные занятия	Практическая подготовка	
1	Web Components: Инкапсуляция на уровне браузера.	8	2		2		4

2	Web Workers: Многопоточность в JavaScript.	8	2		2		4
3	Основы Rust.	16	4		4		8
4	Интеграция Rust и WebAssembly.	8	2		2		4
5	Глубокое погружение в память и Foreign Function Interface.	8	2		2		4
6	Асинхронность и Workers в Rust Wasm.	8	2		2		4
7	WebAssembly SIMD на Rust.	8	2		2		4
8	Оптимизация, отладка и профилирование Wasm.	8	2		2		4
Итого		72	18		18		36

3.2.2 Заочная форма обучения

№ п/п	Разделы/темы дисциплины	Трудоемкость, час					
		Всего	Аудиторная работа				Самостоятельная работа
			Лекции	Семинарские/практические занятия	Лабораторные занятия	Практическая подготовка	
1	Web Components: Инкапсуляция на уровне браузера.	8,5	0,25		0,25		8
2	Web Workers: Многопоточность в JavaScript.	8,5	0,25		0,25		8
3	Основы Rust.	8,5	0,25		0,25		8
4	Интеграция Rust и WebAssembly.	8,5	0,25		0,25		8
5	Глубокое погружение в память и Foreign Function Interface.	8,5	0,25		0,25		8
6	Асинхронность и Workers в Rust Wasm.	8,5	0,25		0,25		8
7	WebAssembly SIMD на Rust.	8,5	0,25		0,25		8
8	Оптимизация, отладка и профилирование Wasm.	8,5	0,25		0,25		8
Итого		72	4		4		64

3.3 Содержание дисциплины

1. Web Components: Инкапсуляция на уровне браузера

Лекция: Custom Elements (жизненный цикл), Shadow DOM (инкапсуляция стилей), `<template>` и `<slot>`. Почему это лучше, чем компоненты в React/Vue для переиспользования на уровне страницы.

2. Web Workers: Многопоточность в JavaScript

Лекция: Архитектура потоков в браузере. Dedicated и Shared Workers. Обмен сообщениями (`postMessage`). Концепция Transferable объектов (передача `ArrayBuffer` без копирования).

3. Основы Rust

Лекция: Базовый синтаксис Rust. Главное отличие от JS/C++ — отсутствие сборщика мусора. Концепции Ownership (владение), Borrowing (заимствование) и Lifetimes (времена жизни). Как типы данных (`String`, `Vec`, `Box`) представляются в линейной памяти.

4. Интеграция Rust и WebAssembly

Лекция: Что такое Wasm под капотом (стековая машина, линейная память). Инструментарий: `wasm-pack`, `wasm-bindgen`. Как генерируются JS-обертки. Передача примитивных типов (числа, булевы значения) между JS и Rust.

5. Глубокое погружение в память и Foreign Function Interface

Лекция: Линейная память Wasm как `ArrayBuffer` в JS. Экспорт и импорт памяти. Передача сложных типов: строк, массивов, структур. Использование `js-sys` и `web-sys` для прямого доступа к DOM и Web API из Rust. Проблема "повреждения указателей" при росте памяти.

6. Асинхронность и Workers в Rust Wasm

Лекция: Почему `std::thread` не работает в браузере. Интеграция Rust Wasm с Web Workers (запуск Wasm внутри воркера). Работа с JS Promise из Rust с помощью `wasm-bindgen-futures` и `async/await`.

7. WebAssembly SIMD на Rust

Лекция: Концепция SIMD (Single Instruction, Multiple data). Wasm SIMD proposal. Векторные типы (`v128`). Использование интринсиков `std::arch::wasm32` в Rust. Ограничения, выравнивание памяти и `endianness`.

8. Оптимизация, отладка и профилирование Wasm

Лекция: Инструмент `wasm-opt` (из Binaryen): уровни оптимизации (`-O1..-O3`, `-Os`), удаление мертвого кода. Source maps и DWARF-информация для отладки Rust-кода прямо в DevTools. Профилирование потребления памяти и поиск утечек на границе JS/Rust.

3.4 Тематика лабораторных занятий

1. Создать кастомный компонент (например, `<custom-slider>` или `<data-grid>`) на чистом JS/HTML/CSS, используя Shadow DOM для изоляции стилей..
2. Вынести тяжелую задачу (например, генерацию фрактала или обработку большого массива данных) в Dedicated Worker. Настроить передачу данных через Transferable для zero-copy..
3. Написать простое консольное приложение на Rust (например, обработку текстового файла или матриц). Скомпилировать его в Wasm-цель..
4. "Hello World" на Wasm. Написать математические функции на Rust, собрать через `wasm-pack`, сгенерировать JS-обертки и вызвать их из HTML-страницы.
5. Написать функцию на Rust, которая принимает массив пикселей (или координаты) из JS, обрабатывает его и возвращает обратно. Написать Rust-код, который напрямую находит DOM-элемент и меняет его стиль через `web-sys`.

6. Перенести тяжелый Rust Wasm модуль из прошлых лаб внутрь Web Worker, чтобы разгрузить главный поток. Реализовать асинхронный вызов: Rust делает fetch запрос (через JS Promise) и обрабатывает полученные данные.
7. Написать функцию для производительной задачи (например, применение фильтра размытия к изображению или перемножение массивов) на скалярном Rust. Затем переписать её с использованием SIMD-интринсиков. Замерить и сравнить скорость в браузере..
8. Взять "раздутый" Wasm-бинарник, уменьшить его размер с помощью wasm-opt. Настроить source maps, чтобы поставить брейкпоинт в исходном Rust-коде из Chrome DevTools. Найти и исправить намеренно оставленную утечку памяти (неочищенный Closure или JsValue).

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

1. Тузовский, А. Ф. Проектирование и разработка web-приложений : учебник для вузов / А. Ф. Тузовский. — Москва : Издательство Юрайт, 2025. — 219 с. — (Высшее образование). — ISBN 978-5-534-16300-1. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/561176> (дата обращения: 22.02.2026).

2. Web-разработки в asp. Net web forms : учебник для вузов / С. Т. Гуляева, В. В. Миронов, Н. О. Котелина, И. И. Лавреш. — Москва : Издательство Юрайт, 2025. — 134 с. — (Высшее образование). — ISBN 978-5-534-19885-0. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/569218> (дата обращения: 11.02.2026).

4.3 Дополнительная литература

1. Лаврищева, Е. М. Программная инженерия и технологии программирования сложных систем : учебник для вузов / Е. М. Лаврищева. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2025. — 432 с. — (Высшее образование). — ISBN 978-5-534-07604-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/561885> (дата обращения: 11.02.2026).
2. Огнева, М. В. Программирование на языке C++: практический курс : учебник для вузов / М. В. Огнева, Е. В. Кудрина, А. А. Казачкова. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2026. — 342 с. — (Высшее образование). — ISBN 978-5-534-18949-0. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/585853> (дата обращения: 11.02.2026).
3. Веб-разработка : учебник для вузов / под общей редакцией О. В. Ратановой, Н. А. Ребус, А. Ю. Анисимова. — Москва : Издательство Юрайт, 2026. — 217 с. — (Высшее образование). — ISBN 978-5-534-21194-8. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/590626> (дата обращения: 11.02.2026).

4.4 Электронные образовательные ресурсы

ЭОР не разработан

4.5 Лицензионное и свободно распространяемое программное обеспечение

1. Microsoft Windows.
2. Веб-браузер, Chrome.
3. Figma
4. Gimp 2.10.20 и выше.
5. Inkscape 1.0 и выше.

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. Лекции должны предшествовать лабораторным работам. Рекомендуемый режим занятий: 2 часа лекций раз в 2 недели и 2 часа лабораторных работ каждую неделю.

2. 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.

3. 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.

В процессе обучения используются следующие оценочные формы самостоятельной работы обучающихся:

- отчёты по лабораторным работам;
- подготовка к зачету;

Отчёты по лабораторным работам проводятся путём предоставления обучающимися самих файлов работы, а также документа-отчёта о выполненной работе с выводами, содержащими анализ полученных результатов. Оценивается выполненная работа баллами от 0-100.

Если отчет представляется позже установленного срока, то за каждую неделю просрочки снимается 10 баллов от максимального, полученного за выполнение работы.

При использовании дистанционной формы обучения в системе LMS предусмотрено итоговое тестирование по теоретическому материалу дисциплины в тест включено 20 тестовых заданий, время на выполнение тестирования составляет 30 минут. Тест считается пройденным в случае правильного ответа на 15 и более тестовых заданий. Примеры тестовых заданий и критерии оценки на экзамене приведены в приложении 2.

Промежуточная аттестация обучающихся в форме зачета проводится по результатам выполнения всех видов учебной работы, предусмотренных учебным планом по данной дисциплине, при этом учитываются результаты текущего контроля успеваемости в течение семестра. Оценка степени достижения обучающимися планируемых результатов обучения по дисциплине проводится преподавателем, ведущим занятия по дисциплине методом экспертной оценки. По итогам промежуточной аттестации по дисциплине выставляется оценка по пятибалльной шкале. К промежуточной аттестации допускаются только студенты, выполнившие все виды учебной работы, предусмотренные рабочей программой по дисциплине «Обработка изображений».

7.2 Шкала и критерии оценивания результатов обучения

Показателем оценивания компетенций на различных этапах их формирования является достижение обучающимися планируемых результатов обучения по данной дисциплине (п. 7.2.1)

7.2.1 Перечень компетенций с указанием этапов их формирования в процессе освоения дисциплины и формы контроля формирования компетенций

Индекс	Компетенция	Форма контроля	Этапы формирования (разделы дисциплины)
ПК-5	ПК-5. Способен осуществлять концептуальное, функциональное и логическое проектирование систем среднего и крупного масштаба и сложности	Промежуточный контроль: Диф. зачет Текущий контроль: проверка лабораторных работ; устное собеседование по результатам выполнения лабораторных работ, промежуточные тесты	1-8

ПК-4	ПК-4. Способен проводить работы по интеграции программных модулей и компонент, и проверку работоспособности выпусков программных продуктов	Промежуточный контроль: Диф. зачет Текущий контроль: проверка лабораторных работ; устное собеседование по результатам выполнения лабораторных работ, промежуточные тесты	1-8
------	--	---	-----

7.3 Оценочные средства

7.3.1 Текущий контроль

Критерии оценки работы обучающегося на лабораторных занятиях

0 баллов

Обучающийся не выполнил лабораторную работу и не предоставил отчет.

1-49 баллов

Обучающийся допустил существенные ошибки при выполнении лабораторной работы и не внес исправления в отчет по лабораторной работе после замечания преподавателя.

50-69 баллов

Обучающийся выполнил лабораторную работу, предоставил отчет вовремя или после указанного срока выполнения. Допускаются неточности в ходе выполнения лабораторной работы, которые были частично исправлены обучающимся после проверки преподавателем.

70-79 баллов

Обучающийся выполнил лабораторную работу, предоставил отчет вовремя или после указанного срока выполнения, допустил неточности, которые были исправлены обучающимся после первой проверки преподавателем.

80-89 баллов

Обучающийся выполнил лабораторную работу, предоставил отчет вовремя. Допускаются незначительные неточности, которые были исправлены обучающимся после первой проверки преподавателем.

90-99 баллов

Обучающийся без ошибок выполнил лабораторную работу, предоставил отчет вовремя.

100 баллов

Обучающийся без ошибок выполнил лабораторную работу, предложил оригинальное решение и предоставил отчет вовремя.

Если отчет представляется позже установленного срока, то за каждую неделю просрочки снимается 10 баллов от максимального, полученного за выполнение работы.

7.3 Оценочные средства

7.3.1 Текущий контроль

Примерный список вопросов

1. Custom Elements (жизненный цикл).
2. Shadow DOM (инкапсуляция стилей), `<template>` и `<slot>`.
3. Почему это лучше, чем компоненты в React/Vue для переиспользования на уровне страницы.
4. Архитектура потоков в браузере.
5. Dedicated и Shared Workers.
6. Обмен сообщениями (`postMessage`).

7. Концепция Transferable объектов (передача ArrayBuffer без копирования).
8. Базовый синтаксис Rust.
9. Главное отличие от JS/C++ — отсутствие сборщика мусора.
10. Концепции Ownership (владение), Borrowing (заимствование) и Lifetimes (времена жизни).
11. Как типы данных (String, Vec, Box) представляются в линейной памяти.
12. Как создать кастомный компонент (например, <custom-slider> или <data-grid>) на чистом JS/HTML/CSS, используя Shadow DOM для изоляции стилей.
13. Как настроить передачу данных через Transferable для zero-copy.
14. Как написать простое консольное приложение на Rust (например, обработку текстового файла или матриц).
15. Как скомпилировать простое консольное приложение в Wasm-цель.
16. Что такое Wasm под капотом (стековая машина, линейная память).
17. Инструментарий: wasm-pack, wasm-bindgen.
18. Как генерируются JS-обертки.
19. Передача примитивных типов (числа, булевы значения) между JS и Rust.
20. Как написать математические функции на Rust, собрать через wasm-pack, сгенерировать JS-обертки и вызвать их из HTML-страницы.
21. Линейная память Wasm как ArrayBuffer в JS.
22. Экспорт и импорт памяти.
23. Передача сложных типов: строк, массивов, структур.
24. Использование js-sys и web-sys для прямого доступа к DOM и Web API из Rust.
25. Проблема "повреждения указателей" при росте памяти.
26. Как написать функцию на Rust, которая принимает массив пикселей (или координаты) из JS, обрабатывает его и возвращает обратно.
27. Как написать Rust-код, который напрямую находит DOM-элемент и меняет его стиль через web-sys.
28. Почему std::thread не работает в браузере.
29. Интеграция Rust Wasm с Web Workers (запуск Wasm внутри воркера).
30. Работа с JS Promise из Rust с помощью wasm-bindgen-futures и async/await.
31. Как перенести тяжелый Rust Wasm модуль из прошлых лаб внутрь Web Worker, чтобы разгрузить главный поток.
32. Как реализовать асинхронный вызов: Rust делает fetch запрос (через JS Promise) и обрабатывает полученные данные.
33. Концепция SIMD (Single Instruction, Multiple data). Wasm SIMD proposal.
34. Векторные типы (v128).
35. Использование интринсиков std::arch::wasm32 в Rust.
36. Ограничения, выравнивание памяти и endianness.
37. Как написать функцию для производительной задачи (например, применение фильтра размытия к изображению или перемножение массивов) на скалярном Rust. Затем переписать её с использованием SIMD-интринсиков. Замерить и сравнить скорость в браузере.
38. Инструмент wasm-opt (из Binaryen): уровни оптимизации (-O1..-O3, -Os), удаление мертвого кода. Source maps и DWARF-информация для отладки Rust-кода прямо в DevTools.
39. Профилирование потребления памяти и поиск утечек на границе JS/Rust.
40. Как "раздутый" Wasm-бинарник, уменьшить его размер с помощью wasm-opt.
41. Как настроить source maps, чтобы поставить брейкпоинт в исходном Rust-коде из Chrome DevTools.
42. Как найти и исправить намеренно оставленную утечку памяти (неочищенный Closure или JsValue).

7.3.2 Промежуточная аттестация

Оценочные средства для промежуточной аттестации не требуется, так как оценка за промежуточную аттестацию выставляется по балльно-рейтинговой системе, описанной в пункте 7.2.