Самостоятельная работа на курсе Web-программирование (2025)

Самостоятельная работа состоит из двух частей, которые оцениваются поровну: сквозной проект и прохождение онлайн-тренажера по вёрстке.

# Онлайн-тренажёр

Для самостоятельного обучения базам веб-разработки, мы предлагаем пройти бесплатные курсы-тренажёры от Hexlet

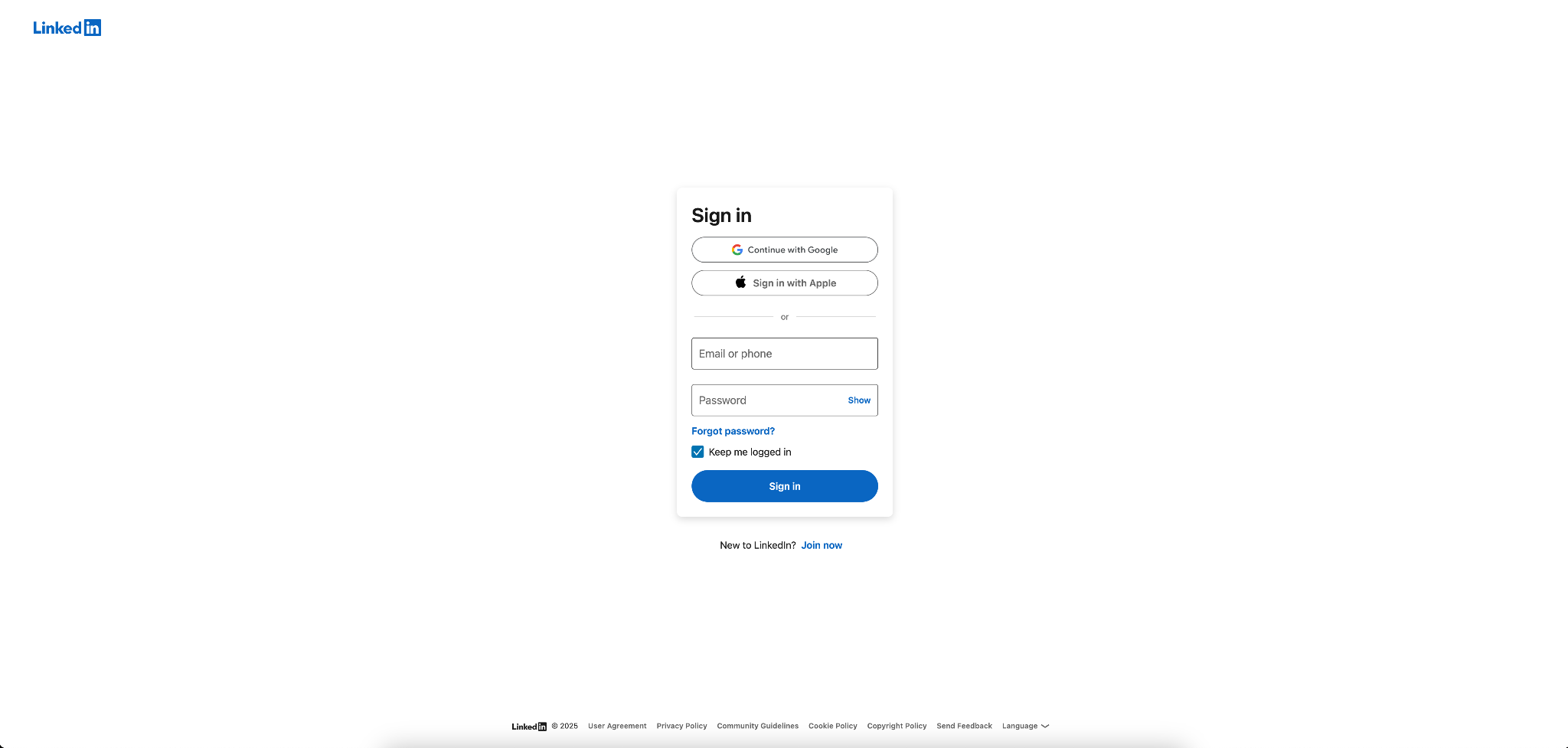
* HTML (41 балл): <https://code-basics.com/languages/html>
* CSS (38 баллов): <https://code-basics.com/languages/css>
* JavaScript (72 балла): <https://code-basics.com/languages/javascript>

Итоговый балл — процент выполнения курсов. Мы суммируем все ваши баллы и считаем процент от максимума (max=151)

Для сдачи проекта необходимо продемонстрировать прохождение курса на созвоне / лично, скриншот не подойдёт.

# Сквозной проект: разработка интерактивной формы авторизации

Итоговая оценка за сквозной проект — процент от максимального балла + бонусы за дополнительные задания и усложнения.



## Создание репозитория на GitHub

* Форма должна быть в публичном репозитории на GitHub. У репозитория должен быть оформлен **README**, в нем указаны **инструкции** по запуску, **ответы на открытые вопросы со значком ❓**
* Проект будет состоять из трёх частей: нативные технологии, нативные технологии, но с инфраструктурой, разработка с фреймворком.
* Предлагается выделить три папки внутри репозитория: native, with-infrastructure, framework. Разработка всех трех частей ведётся независимо, но для всех последующих этапов рекомендуется скопировать наработки предыдущих этапов
* На всех трёх этапах форма должна выглядеть и работать аналогично

## «Нативные» технологии (27+ баллов)

Сначала сделаем всё «по старинке», чтобы понять как в жизни бывает.

### HTML (10 баллов)

Используя технологии HTML, разработайте форму авторизации, содержащую:

* «Шапку» и «подвал» страницы **(1 балл)**
* Форму логина, состоящую из кнопок входа через соцсети (1 б), полей для логина и пароля (1 б), галочки «сохранять сессию» (1 б), кнопки входа (1 б), итого **4 балла**
* Использовать семантические теги **(1 балл)**
* Сделать форму доступной для людей с ограниченными возможностями (accessibility), ❓объяснить как сделали **(3 балла)**
* ❓Необходимо обосновать использование подхода к верстке (блочная, flex, grid) **(4 балла)**

### CSS (8+ баллов)

* Улучшить стили, не использовать стили нативных HTML-элементов (переопределить стили для всех заголовков, текста, контролов) **(1 балл)**
* Добавить transition-анимацию на поля ввода/кнопки (примеры: подсветка некорректного ввода, загрузка после нажатия конпки “логин”, переливающийся градиент блока) **(1 балл)**
* Добавить keyframe-анимацию (примеры: анимация фона окна, загрузка кнопки, “сворачивание” окна, “дрожащий” инпут на неверный ввод) **(1 балл)**
* Использовать media queries для адаптивной вёрстки, ❓объяснить выбор брейкпоинтов **(4 балла)**
* При работе со стилями использовать:
  + Селектор по классу **(1 балл)**
  + Селектор по атрибуту **(1 балл)**
  + Селектор по идентификатору **(1 балл)**
  + Селектор по тегу **(1 балл)**
  + 3-4 [псевдоселектора](https://developer.mozilla.org/ru/docs/Web/CSS/Pseudo-classes) по классу **(1 балл)**
* Повеселиться и добавить снежинки/конфетти на фон, пасхалку и т.п (доп. баллы)

### JS (9+ баллов)

* Использовать js для валидации полей ввода — установить ограничения пароля, почты или телефона **(1 балл)**
* Использовать js для условного вывода сообщений пользователю **(1 балл)**
* Использовать js для проверки правильности введённых данных (захардкодить верную комбинацию данных и сравнить ввод) **(1 балл)**
* Использовать js для сохранения введённых данных, их последующего отображения ❓ объяснить почему выбрали тот или иной способ хранения данных, помним, что это пара логин-пароль **(4 балла)**
* Использовать обработчики событий **(1 балл)**
* Использовать стрелочные и именованные функции ❓ поделиться, какие больше понравились, почему? **(1 балл)**
* Повеселиться и добавить снежинки/конфетти на фон, пасхалку и т.п (доп. баллы)

## Подключаем инфраструктуру (24 балла)

Необходимо взять предыдущий проект и прокачать его инфраструктуру. Предыдущий проект (нативный) менять нельзя — разделите проект на несколько папок, чтобы было наглядно видно как проект менялся.

Обратите внимание, что проект должен запускаться под unix-системами в первую очередь. Для этого, вместо командной строки или PowerShell может быть удобно использовать терминал git bash / MINGW64

### Подключите менеджер пакетов (3 балла)

Используйте менеджер пакетов: npm, yarn, pnpm или другой. ❓ Объясните свой выбор.

⚠️ Для создания «проекта» из папки, как правило, менеджеры пакетов предоставляют команду init.  
 ⚠️ Версию пакетного менеджера можно дополнительно зафиксировать в секции engines файла package.json

### Зафиксируйте версию nodejs с использованием nvm или аналога (2 балла)

Используйте утилиту nvm (или аналог) для фиксации версии nodejs.  
 ⚠️ Версию node можно дополнительно зафиксировать в секции engines файла package.json

### Подключите Typescript (4 балла)

Установите в проект Typescript. Постарайтесь сконфигурировать проверку типов максимально строго.

* Сконфигурируйте tsconfig.json, ❓объясните выбор конфигурации. Выбор готовой конфигурации — это тоже выбор и его можно объяснить.
* Опишите типы данных вашего приложения.
* Добавьте в package.json команду для проверки типов (tsc --noEmit)

### Подключите сборщик (4 балла)

Подключите к вашему проекту сборщик и дев-сервер.

* Опишите команды serve и build для запуска дев-сервера и сборки проекта.
* ❓ Объясните выбор сборщика.

### Подключите линтер (3 балла)

* Установите пакет линтера и необходимые плагины
* ❓объясните выбор конфигурации. Выбор готовой конфигурации — это тоже выбор и его можно объяснить.
* Опишите команды lint и lint:fix для проверки качества кода и автоисправления ошибок.

### Подключите форматтер (3 балла)

* Установите пакет форматтера и необходимые плагины
* ❓объясните выбор конфигурации. Выбор готовой конфигурации — это тоже выбор и его можно объяснить.
* Опишите команду format для проверки форматирования кода и автоисправления ошибок.

### Подключите фреймворк тестирования (5 балла)

* Установите пакет фреймворка тестирования (юнит-тесты)
* Сконфигурируйте фреймворк тестирования
* ❓объясните выбор фреймворка и конфигурации
* Напишите тесты

## Используем фреймворк (20 баллов)

В этой части сделаем всё снова, но используем React

Необходимо взять предыдущий проект и натянуть на него фреймворк. Предыдущие проекты (нативный, с инфраструктурой) менять нельзя — разделите проект на несколько папок, чтобы было наглядно видно как проект менялся.

Обратите внимание, что проект должен запускаться под unix-системами в первую очередь. Для этого, вместо командной строки или PowerShell может быть удобно использовать терминал git bash / MINGW64

### Подключите React и инфраструктуру (5 баллов)

* Установите React актуальной версии
* Используйте знания предыдущей части, чтобы подключить инфраструктуру (ts, линтер, сборщик). МОЖНО использовать готовые фреймворки и бойлерплейты с явным указанием и обоснованием выбора того или иного варианта

### Подключите стейт-менеджер (5 баллов)

* Выберите стейт-менеджер, либо подробно обоснуйте почему он не нужен (да, за это баллы тоже можно получить)
* Используйте стейт-менеджер в вашем проекте

### Реализуйте интерфейс из предыдущей части (5 баллов)

* Разбейте интерфейс на переиспользуемые компоненты
* Используйте компоненты для построения интерфейса
* Можно использовать готовые библиотеки компонентов (Ant Design, аналоги)

### Протестируйте компоненты (5 баллов)

* Установите библиотеку для тестирования компонентов, обоснуйте свой выбор
* Покройте проект тестами

### Дайте волю фантазии на дополнительные баллы

* Сделайте вызовы к несуществующему апи
* Сделайте роутинг и несколько страниц
* Сделайте крутые анимации
* Что угодно прикольное можно делать…