Методология курса Основы Web-разработки

Цель и задачи занятия

Цель

- Ознакомиться с методологией курса

Задачи

- Получить представление об обычном рабочем процессе программиста в рамках организации
- Узнать, как структура практических занятий связана с реальными практическими ситуациями
- Понять и применять впоследствии рекомендации по выполнению практических заданий

Программист – это практик

- Технологии развиваются быстрее, чем выходят учебные материалы
- Лучшая теория – это теория полученная практическим путем
- Хороший технический руководитель управляет на «экспертности»
- Хорошие IT менеджеры (тим-лиды, аналитик итп) разбираются в технологиях

Хорошо что Это НЕ IT =)

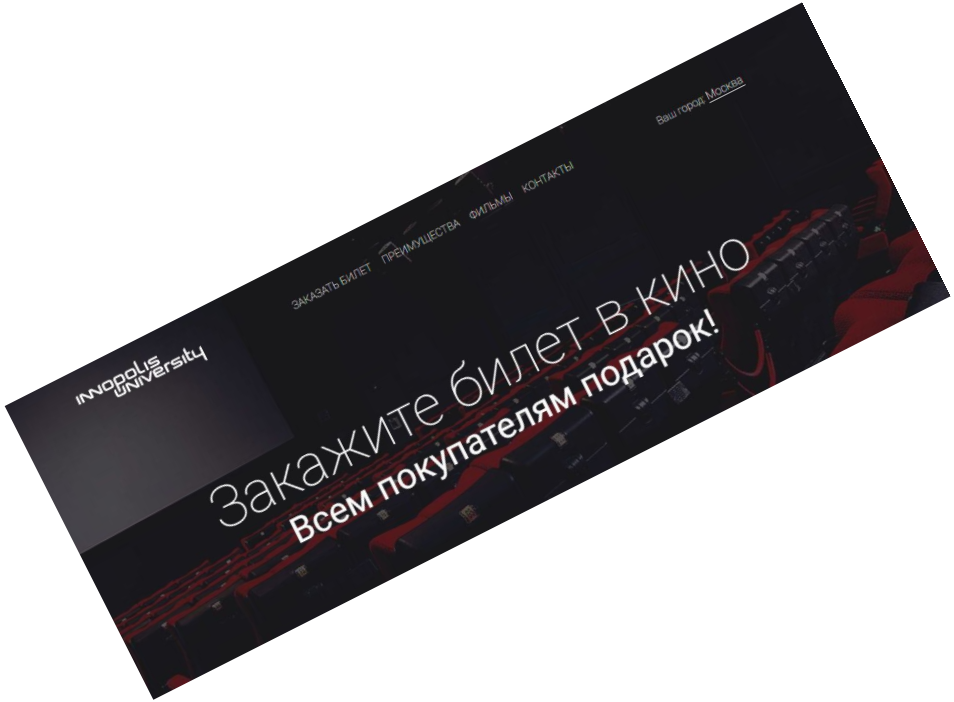
Практико-ориентированный курс

Цель

- Создать веб-сайт (лэндинг-пэйдж) для кинотеатра с возможностью бронирования билетов

Задачи

- Получить макет в формате *.psd
- Ознакомиться с учебным примером
- Ознакомиться с техническим заданием
- Сделать в рамках технического задания самостоятельно



Критерии успешного завершения курса

- **Зачет по теории.** Итоговый тест – 65% верных ответов. Проводится до Зачета по практической части

- **Зачет по практике.** Приёмка лэндинга, сделанного на курсе, в соответствии с техническими требованиями. Приёмка осуществляется преподавательским составом.

- **Зачет по практике (альтернатива).** Не менее 21 балла за проделанную практическую работу

Практико-ориентированный курс

Особенности

- Теоретические аспекты разбираются в процессе практической деятельности или минимум теории
- Как можно раньше использовать получаемые теоретические знания на практике и писать код
- Самостоятельные и семинарные задания курса являются составными частями выпускного проекта
- Погружение слушателя в условия, максимально приближенные к реальным рабочим условиям
- Каждая практическая тема – это релиз (выпуск) программного продукта

Практико-ориентированный курс

Практические работы построены по системе релизов ПО

- Релиз - законченная версия продукта. Промежуточная или основная
- На курсе подкрепляется двумя документами PD и CA
- Названия документов по их разделам (Plan, Do, Check, Act)

Рабочий процесс программиста

- Самодисциплина
- Анализ задачи и оценка
- Исполнение задачи
- Промежуточное тестирование
- Доработка
- Сдача (code-review)

Техническое задание

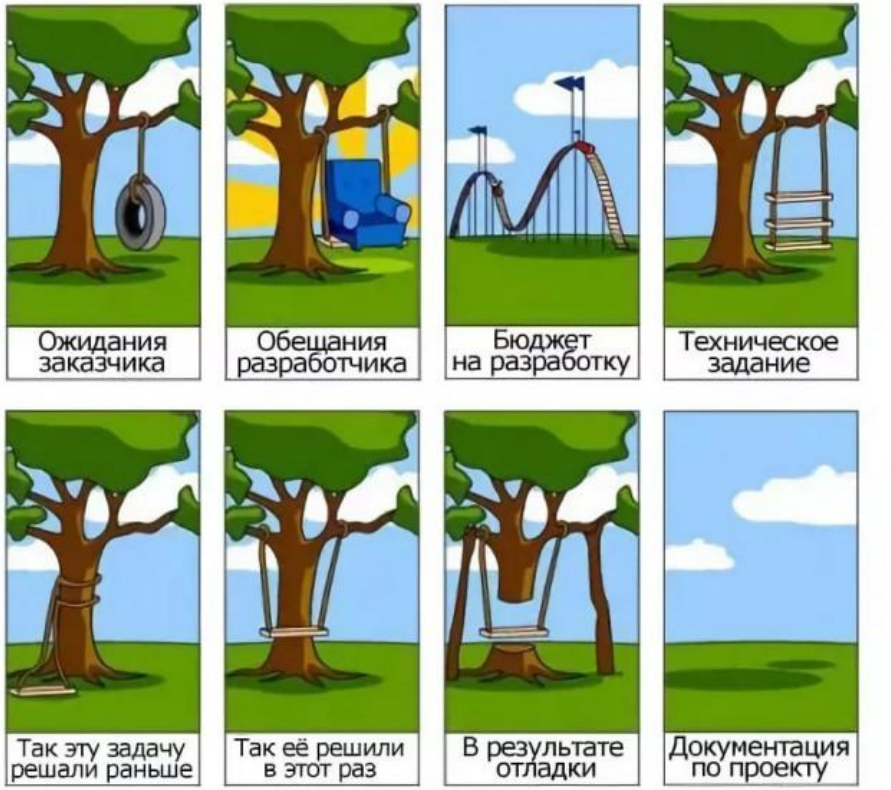
Техническое задание – это строго формализованный документ, оформленный в соответствии с внутренними стандартами в организации или отраслевыми стандартами, максимально полно описывающий поведение программной системы, ее функционал, ограничения и требования к внешней среде ПО (люди, оборудование, сети, стороннее ПО и.т.д.).

Возможные стандарты: ГОСТ 34, ГОСТ 19, IEEE 29148-2011 и др.

Анализ задачи

Задачи ставятся плохо!

- понять контекст задачи (зачем ?)
- понять полноту условий (все ли описано?)
- понять однозначность трактовки задачи
- оценить срок исполнения
- задать уточняющие вопросы
- приступить к исполнению



Функциональные требования

Функциональные требования – это описание того, как программное обеспечение ведет себя при определенных действиях пользователя. Какую полезную работу выполняет. Получению каких результатов служит.

- Use Cases – пользовательские истории
- Формализованное описание

Нефункциональные требования

Нефункциональные требования – это прочие требования, не относящиеся к поведению системы в целом. Это могут быть требования, например, к используемым технологиям. Требования к выдерживаемым нагрузкам. Дополнительные описания по взаимодействию с другими частями системы. Либо какие-то ограничения.

Анализ задачи

Договориться!

Ваши функциональные требования на курсе

Раздел Plan – ваши требования!

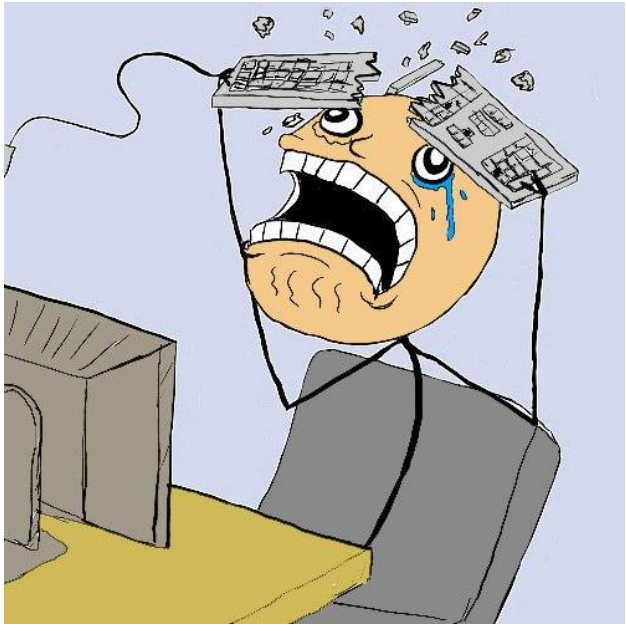
Исполнение задачи

Готовые сторонние решения и «копипаста»

- *я знаю готовое решение и где его взять*
- *сейчас «по-быстрому» его прикрутим и все будет хорошо работать*

Исполнение задачи. Готовые решения

- Срыв сроков
- Повышение сложности кода
- Несоответствие требованиям задачи



Исполнение задачи. Готовые решения

- Актуальность и уместность решения
- Используемые технологии
- Возможности кастомизации
- Документация
- Отраслевые и организационные стандарты
- Рекомендации

Хорошая копипаста – это копипаста с умом !

Самое главное понимать с чем работаешь!

Исполнение задачи. Кривая обучения

- В каждом проекте что-то новое
- Новая технология требует изучения
- Учишься самостоятельно
- **Учишься у коллег**
- **Учишься у сообщества**

Самое главное понимать с чем работаешь!

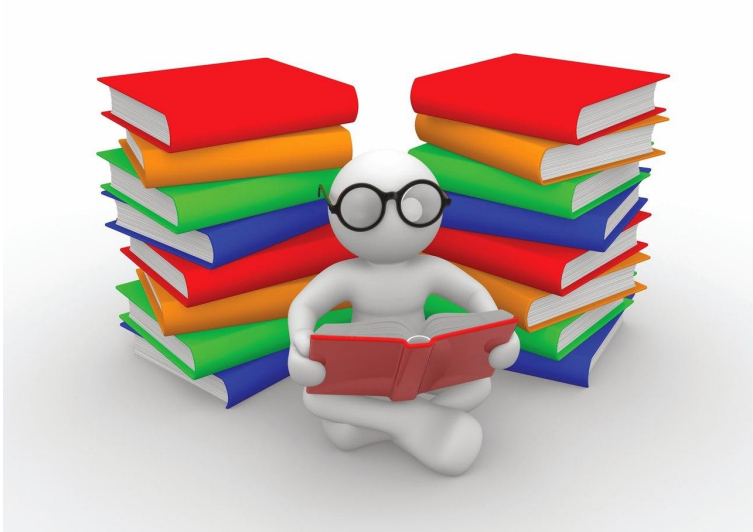
Ваши функциональные требования на курсе

Раздел DO – ваша шпаргалка!

А семинар – это ваши старшие коллеги

Промежуточное тестирование

- На следующий день после выполнения семинарного занятия вернуться к программным требованиям
- Проверить, все ли было выполнено на семинарном занятии в соответствии с разделом PLAN
- Проверить все ли вы поняли из семинарного занятия (раздел Check)
- Выписать возникшие вопросы и недочеты



Промежуточное тестирование

Раздел CHECK – ваш самоконтроль и фиксация знаний!

Доработки

У заказчика всегда есть требования и новые задачи!

Виды возникающих задач:

BUGFIXES – исправление выявленных ошибок и недочетов в соответствии с требованиями

FEATURES – новый функционал, не обсуждавшийся ранее

REFACTORING – исправление существующего кода

Доработки

Раздел ACT must – это требования вашего строгого заказчика и все должно быть сделано на отлично !

Консультации

1. **Приёмка Релиза.**

Консультант – суровый заказчик

2. **Ответы на вопросы**

Сдача (code-review)

- После выполнения раздела Act-Must еще раз самостоятельно проверить соответствие требованиям релиза
- Сделать коммит в свой **основной** репозиторий проекта и прикрепить ссылку на него в Moodle; а так же указать наименование последнего рабочего коммита
- Правила именования коммитов

<номер темы>-<номер коммита в теме по порядку>-<комментарий>

- **Допуском к приёмке работы является условие, что семинарная часть (раздел DO), домашняя часть (раздел ACT) – сделаны; коммит сохранен в ваш репозиторий проекта и номер коммита размещен в системе Moodle**
- Созвониться с консультантом в установленное время
- Проводится либо И приёмка работы И консультация. Либо только консультация

Сдача (code-review) приёмка на консультации

- Слушатель коротко (2-3 мин.) рассказывает о проделанной работе и примененных в работе знаниях (приобретенных на семинаре и при самостоятельной работе)
- консультант убеждается, что релиз соответствует функциональным и нефункциональным требованиям всего релиза (раздел PLAN)
- консультант выносит рекомендации по оформлению кода или подходам, применяемым в решении задачи; а так же отвечает на вопросы слушателя
- слушатель фиксирует полученные рекомендации и ответы на вопросы
- консультант принимает решение о принятии работы. Необходимым условием для принятия работы является соответствие требованиям. В противном случае, работа возвращается на доработку

Сдача (code-review)

Система оценки домашних заданий

✔️а) Код работает в соответствии с техническим заданием (главный критерий) = **0.5 балла**
✔️б) Код правильно оформлен (оценивается только при выполнении критерия а) = **0.5 балла**
✔️в) Выполнено дополнительное задание и его код работает (оценивается только после выполнения критериев а и б) **= 0.5 балла**

Таким образом:
Max. балл за одно задание = 1,5 балла.
Ваша цель: набрать за каждое ДЗ по критериям “а)код работает” + “б)код правильно оформлен” = 1 балл.

Всего на курсе 42 задания:
- Максимум можно набрать 63 балла (42 задания*1,5 балла)
- Ваша цель: набрать 42 балла (42 задания*1 балл)
- Для зачета достаточно будет набрать в сумме 21 балл (42 задания*0,5 баллов)с

Что если проект не принят?

1. Скопировать пример проекта предыдущего занятия
2. Продолжить работать совместно с группой на семинарных занятиях на основе примера
3. Нагнать «учебный проект»

Рекомендации по оформлению кода:

- верстка выполнена в соответствии с макетом psd; pixel perfect с допуском не более 5px
- при выборе селекторов (верстка) отдается предпочтение классам перед id; с уровнем вложенности не более 2
- при выборе селекторов (JavaScript функциональность) отдается предпочтение id
- именование классов CSS осмысленное, соответствует логической структуре документа, на английском языке, соответствует методологии БЭМ
- программный код PHP и JavaScript логически структурирован. Необходимо применять там, где нужно, объектно-ориентированный подход.
- грамотное разделение на классы, методы. Названия переменных, классов и методов должны быть осмыслены и отражать их поведение (предназначение).
- именование классов, методов, переменных в PHP коде соответствует стандартам PSR-2, PSR-4 (https://geekbrains.ru/posts/php_code_conventions)
- именование классов, методов, переменных в JS коде соответствует рекомендациям Google JavaScript Style Guide, а также https://learn.javascript.ru/coding-style

Вопросы?

@wslapshin