

ОСНОВНАЯ ПРОФЕССИОНАЛЬНАЯ ОБРАЗОВАТЕЛЬНАЯ ПРОГРАММА ПОДГОТОВКИ БАКАЛАВРА

по направлению

09.03.01 Информатика и вычислительная техника профиль

"Технологии разработки программного обеспечения"

Б. 1.10.1 Модуль "Проектирование и разработка веб-решений".

Веб-проектирование и веб-языки

Приложение 1

Типовые задания для проведения процедур оценивания результатов освоения дисциплины

в ходе текущего контроля, шкалы и критерии оценивания

Содержание

1. [Типовые задания лабораторных работ по темам](#)
2. [Типовые задания для инвариантной самостоятельной работы по темам](#)
3. [Типовые задания для вариативной самостоятельной работы по темам](#)

1. Типовые задания лабораторных работ по темам

Задания лабораторных работ направлены на отработку действий, способов, методов решения задач в области веб-проектирования и веб-дизайна, которыми обучающиеся должны владеть.

Задания для лабораторных работ предполагают преимущественно репродуктивный характер действий обучающихся и рассчитаны на выполнение в рамках определенного количества часов лабораторных занятий (Таблица 3). Задания представлены в СДО Moodle в электронном учебном курсе по дисциплине «Веб-проектирование и веб-языки» и могут иметь одну или несколько из следующих форм:

- краткая формулировка постановки задачи с пояснениями;
- пошаговая инструкция;
- скринкаст.

Далее приводятся ссылки на веб-страницы на сервере Кодактор.ру (например, https://kodaktor.ru/g/xml_intro), содержащие более детальные инструкции к заданиям и поясняющие скринкасты.

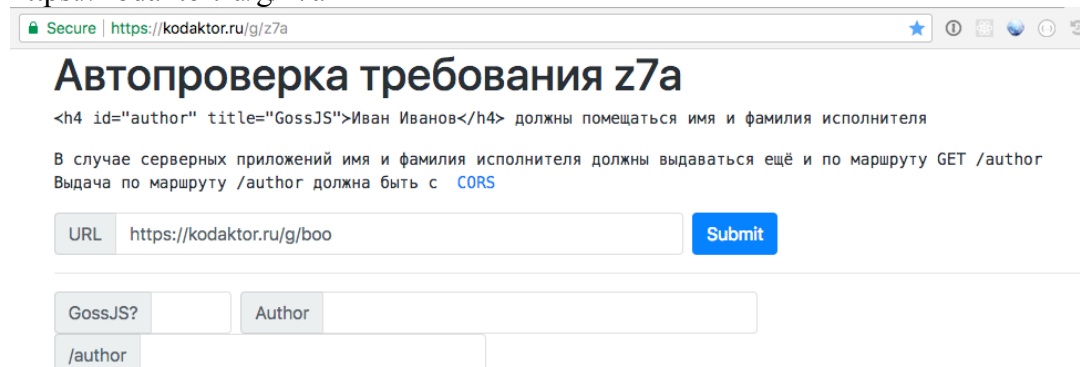
Выполнение задания предполагает следующие виды деятельности:

- разработку сценария или приложения на языке разметки XML или веб-языке (языке программирования высокого уровня в области веб-ресурсов):
 - JavaScript;
 - и, возможно, другом языке по выбору обучающихся;
- выявление и исправление синтаксических ошибок, выполнение транспиляции, тестирования работы сценария.
- составление отчета о выполненном задании в виде слайдов и/или онлайн-документации, размещаемой в Git-репозитории как части веб-портфолио обучающегося (https://github.com/GossJS/report_template и https://kodaktor.ru/g/report_template).

Критерий оценивания. Лабораторная работа считается выполненной, если XML-документ или веб-сценарий разработан и соответствует заданию, не содержит синтаксических ошибок, а также сопровождается репозиторием (в том числе, возможно,

отчётом в форме слайдов). Часть заданий проверяется (или дополнительно проверяется) с помощью средств автоматизированной проверки, отсылающей запросы к веб-сценариям. Веб-сценарии должны быть размещены на ресурсе, допускающем обращение через Интернет (онлайн-редактор типа Кодактор.ру или codepen.io):

<https://kodaktor.ru/g/z7a>



Secure | <https://kodaktor.ru/g/z7a>

Автопроверка требования z7a

<h4 id="author" title="GossJS">Иван Иванов</h4> должны помещаться имя и фамилия исполнителя

В случае серверных приложений имя и фамилия исполнителя должны выдаваться ещё и по маршруту GET /author
Выдача по маршруту /author должна быть с [CORS](#)

URL

GossJS? Author

/author

Выполнение заданий самого простого вида в Кодактор.ру: <https://kodaktor.ru/dzjs>

Тема 1. Языковые средства веб-технологий на основе XML и CSS и развёртывание среды разработки компонентов аппаратно-программных комплексов.

Разработка предметного языка разметки на основе XML (https://kodaktor.ru/g/web_intro и https://kodaktor.ru/g/xml_intro)

1. Представьте свои текущие знания в области HTML в виде набора тегов (правильно сформированного XML-документа)
2. Осуществите рефакторинг DTD для данного документа, уменьшив количество повторяющихся инструкций
3. Рассмотрите документ [Статистика посещений страницы](https://kodaktor.ru/g/08092017_stats) (https://kodaktor.ru/g/08092017_stats). Основываясь на нём, разработайте язык для описания посещений некоторого URL, т.е. создайте DTD и валидный документ.

Отразите в нём:

- самый частый IP-адрес
- количество посещений со стационарных платформ:
 - Windows
 - Linux
 - MacOS
- количество посещений с iPhone и других мобильных платформ

создайте раздел IP-адресов <ips></ips> в котором повторяющиеся тэги <ip addr="" frequency=""/>

описывают пять самых часто встречающихся адресов в порядке убывания

5. Разработайте или сгенерируйте схему (XML Schema) для документа:

```
<?xml version="1.0" encoding="utf-8" ?>
<таблица>
  <студент имя = "Иванов">
    <отметка дисциплина="Мультимедиа">3</отметка>
    <отметка дисциплина="Веб-дизайн">3</отметка>
    <отметка дисциплина="Графика">5</отметка>
  </студент>
  <студент имя = "Петров">
    <отметка дисциплина="Мультимедиа">3</отметка>
    <отметка дисциплина="Веб-дизайн">4</отметка>
    <отметка дисциплина="Графика">5</отметка>
    <отметка дисциплина="Логика">5</отметка>
  </студент>
  <студент имя = "Сидоров">
    <отметка дисциплина="Мультимедиа">3</отметка>
    <отметка дисциплина="Веб-дизайн">4</отметка>
    <отметка дисциплина="Графика">5</отметка>
    <отметка дисциплина="Логика">4</отметка>
  </студент>
</таблица>
```

и осуществите её валидацию с помощью инструмента типа xmlvalidation.com

6. Разместите полученный отчёт (лог действий) в репозитории (веб-портфолио).

Тема 2. Основы современного JavaScript (ECMAScript 2016, ECMAScript 2017): синтаксис, структуры данных и функциональный стиль программирования

I. Развёртывание и настройка папки проекта и приложений для управления JavaScript-проектом и мониторинга зависимостей

1. Убедитесь что установлены node, npm и yarn (https://kodaktor.ru/js01_intro_lr.pdf)
2. Создайте новый проект: `mkdir $(date +%Y%m%d_%H%M%S) && cd $_ && yarn init -y` или `mkdir $(date +%Y%m%d_%H%M%S) && cd $_ && npm init -y` (<https://kodaktor.ru/g/init>)

3. Добавьте пакет `diff2html-cli` как девелоперскую зависимость: `yarn add --dev diff2html-cli` или `npm i -D diff2html-cli`

Когда зависимость успешно установлена, то запись об этом добавляется в файл `package.json`:

```
"devDependencies": {
  "diff2html-cli": "^2.5.4"
}
```

4. Установите инструмент `nodemon` для автоматизации перезапуска сценария: `yarn add --dev nodemon` или `npm i -D nodemon`
5. Поместите каждую из строк

```
console.log('Hello, world!');
console.log('Hello, %s!', 'world');
```

в отдельный файл (`1.js` и `2.js`) и выполните команду:

```
git diff --no-index 1.js 2.js | diff2html -d word -s line -i stdin
```

Результат можно сверить с тем, что размещён по адресу <https://kodaktor.ru/g/diff1>

```

1.js → 2.js [RENAME]
@@ -1 +1 @@
1 - console.log('Hello, world!');
1 + console.log('Hello, %s!', 'world');

```

6. Запустите любой из сценариев с помощью nodemon: `npx nodemon ./1.js` и убедитесь, что выводится ожидаемая надпись.
7. Измените надпись на любую другую и сохраните изменение в файле. Убедитесь, что результат мгновенно отразился на выводе.
8. Разместите полученный отчёт (лог действий) в репозитории (веб-портфолио).

II.

Часть А. Разработка веб-сценария, содержащего решение линейной системы уравнений методом Крамера:

1. Создайте веб-страницу (борд) на Кодактор.ру или ином онлайн-редакторе
2. Объявите переменные, соответствующие коэффициентам уравнений:
`const [a1, b1, c1, a2, b2, c2] = [1, 3, 2, 9, 2, 7];`
3. Напишите три инструкции, помещающие определители в переменные `det`, `det1` и `det2`
4. Напишите две инструкции, помещающие решение в переменные `x` и `y`
5. Напишите инструкцию, выводящую результат в консоль.
6. Разместите адрес борда (а также его скриншот и лог действий) в репозитории (веб-портфолио) и в специально созданном форуме в соответствующем курсе в СДО Moodle.

Часть Б. Разработка веб-сценария в функциональном стиле JavaScript с использованием условного (тернарного) оператора

1. Пусть есть выражение, генерирующее случайное число от 0 до 100. Используем тернарный оператор для того, чтобы идентифицировать, выпало ли число 50, находящееся в середине этого диапазона:

```

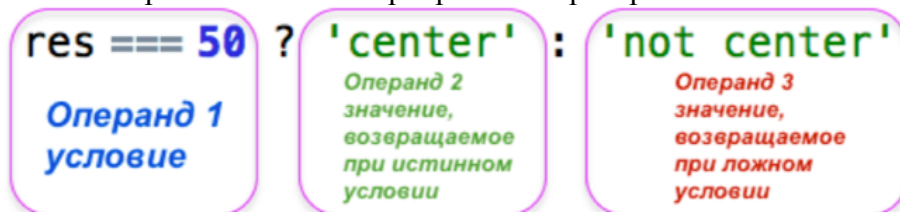
const
  a = 0,
  b = 100,
  res = Math.floor(a + Math.random() * (b - a + 1));

const middle = res === 50 ? 'center' : 'not center';

console.log(middle);

```

2. Рассмотрите синтаксис тернарного оператора:



4. Рассмотрите код в строках 11-13 на странице <https://kodaktor.ru/ternary>

```

const age = 17;
const restricted = ( age < 18 ) ? 'yes' : 'no' ;
Out.log( restricted );

```

5. Доработайте его так, чтобы переменная `restricted` принимала не одно из двух, а одно из трёх различных значений: (а) значение `yes` при значении переменной `age` меньше 18 (б) значение `notsure` при значении переменной `age` равном 18 (в) значение `no` в противном случае

Часть Б.

6. Так как в JavaScript существуют значения, которые нестрого равны друг другу при неявном приведении типов к логическому (они приводятся к `false` и называются `falsy`, «ложностные»), а одно из этих значений ещё и не равно самому себе, то нужен способ отличать их друг от друга. Функция `isNaN` тоже занимается неявным приведением:

```
> isNaN()  
true  
> isNaN('p')  
true
```

(при этом отметим, что `Math.sqrt(-1)` не приводится к `NaN`, а в точности есть `NaN`, так же как литерал значения `NaN`, выглядящий в программе как `NaN`).

С использованием операторов напишите тернарный оператор, возвращающий:

'=NaN', если тестируемое значение в точности есть `NaN`,

'=null', если тестируемое значение в точности есть `null`,

'=undefined', аналогично,

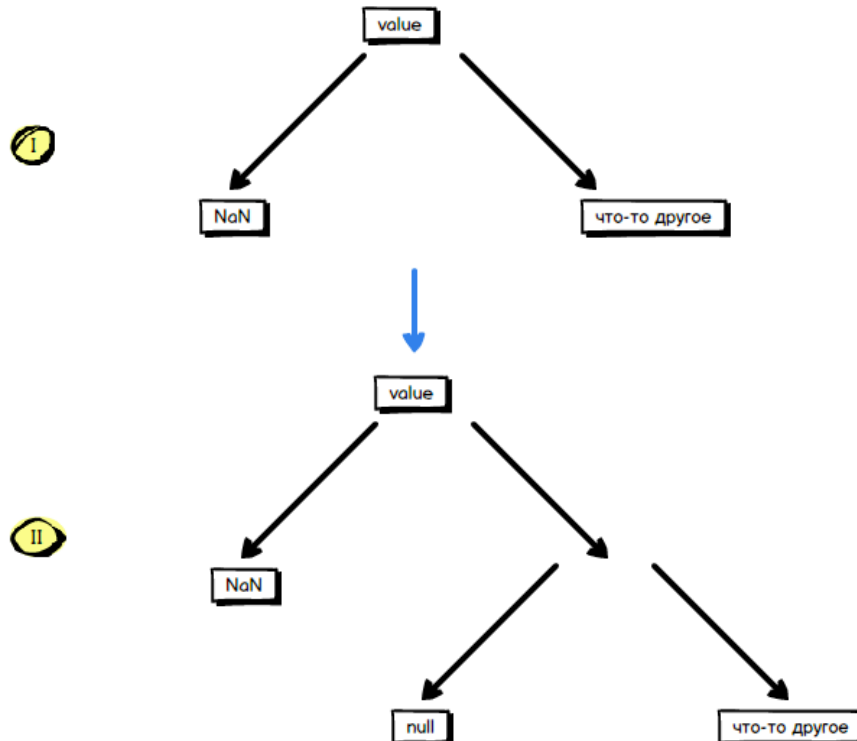
'=0', аналогично.

'=', в случае пустой строки

и

'=false' в случае значения `Boolean False`.

Для этого поэтапно спроектируйте дерево вида



Тема 3. Элементы асинхронного и событийно-ориентированного клиентского веб-программирования.

I.

1. Создайте веб-страницу (борд) на Кодактор.ру или ином онлайн-редакторе
2. Ознакомьтесь с инструкцией и скринкастом по адресу https://kodaktor.ru/g/json_intro
3. Осуществите линтинг JSON-документа (список валют) по адресу <https://kodaktor.ru/j/rates> с помощью jsonlint.com и сгенерируйте для него схему с помощью запроса <https://kodaktor.ru/api/schema/rates>, после чего отвалидируйте этот документ относительно этой схемы любым доступным способом
4. Напишите в качестве бета-версии выполнения этого задания инструкции по образцу <https://kodaktor.ru/logins09012018> которые выводят список названий валют извлечённых из указанного выше списка. Используйте глобальный метод `fetch`, а также методы `createElement` и `appendChild`.
5. Напишите инструкции, улучшающие предыдущий вариант, которые представляют курсы валют в виде таблицы:

Валюта	Продажа	Покупка
Доллар США	69.55	67.26
Евро	77.76	75.14
Фунт	98.02	94.77
Франк	71.27	68.86
Юань	10.70	10.33
Злотый	18.25	17.64

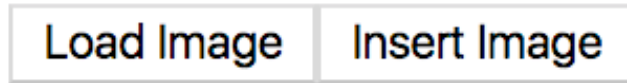
6. Разместите адрес борда (а также его скриншот и лог действий) в репозитории (веб-портфолио) и в специально созданном форуме в соответствующем курсе в СДО Moodle.
- #### II. Разработка веб-сценария, формирующего галерею кэшированных изображений, с помощью модификации дерева DOM

1. Создайте веб-страницу (борд) на Кодактор.ру или ином онлайн-редакторе
2. Напишите функцию, которая создаёт и возвращает элемент `img` по переданному ей адресу изображения (`url`):

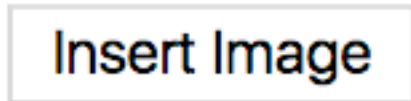
```
function cacheIm(url){  
    const im = document.createElement('img');  
    im.src = url;  
    return im;  
}
```

3. Напишите инструкцию, которая вставляет рисунок, возвращённый этой функцией: `document.body.appendChild(cacheIm('http://www.domaingyan.com/wp-content/uploads/2012/07/web-hosting1.png'))`;
4. Добавьте обработчик событий таким образом, чтобы по щелчку по этому изображению оно обводилось толстой красной рамкой.

5. Пусть есть массив адресов изображений. Используйте метод `forEach` для вставки каждого из этих изображений подряд на веб-страницу.
6. Пусть описанное выше выполняется по щелчку по кнопке. Используйте метод `addEventListener` для добавления события `click`



Нажатие на вторую кнопку не должно производить никаких действий, пока не нажата первая кнопка. После нажатия первой кнопки изображение загружается, а сама первая кнопка исчезает.



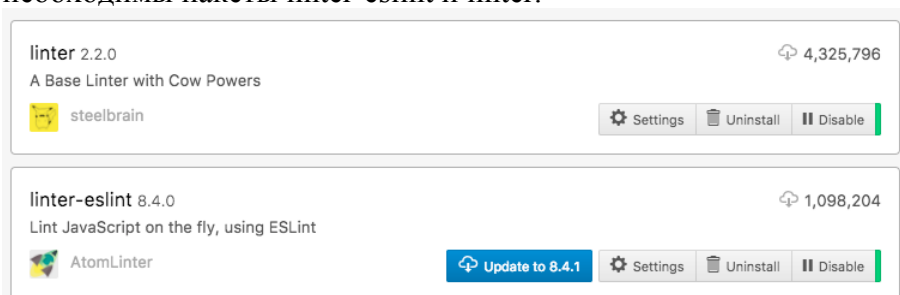
Далее после нажатия по второй кнопке вставляется загруженное ранее изображение.

7. Разместите адрес борда (а также его скриншот и лог действий) в репозитории (веб-портфолио) и в специально созданном форуме в соответствующем курсе в СДО Moodle. Образец: https://kodaktor.ru/dom_imgins

Тема 4. Экспериментальная проверка корректности функционирования (тестирование) веб-приложений

I. Установка и настройка линтера кода на языке JavaScript

1. Создайте новый проект: `mkdir $(date +%Y%m%d_%H%M%S) && cd $_ && yarn init -y` или `mkdir $(date +%Y%m%d_%H%M%S) && cd $_ && npm init -y` (<https://kodaktor.ru/g/init>)
2. Убедитесь, что в используемом вами редакторе кода установлен нужный плагин для работы с линтером `eslint`: например, для работы с редактором `Atom` необходимы пакеты `linter-eslint` и `linter`.



3. Установите в проект настройки линтера `eslint-config-airbnb`. На странице пакета `eslint-config-airbnb` предлагается команда, работающая в Linux/macOS, которая производит все нужные действия:

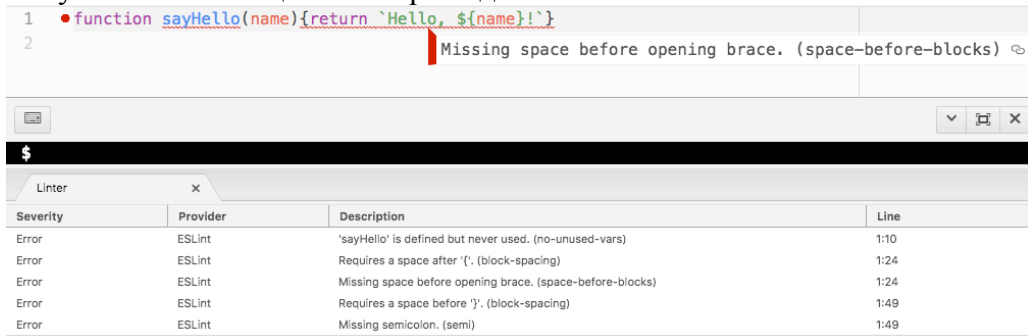
```
(
  export PKG=eslint-config-airbnb;
  npm info "$PKG@latest" peerDependencies --
  json | command sed 's/[\\}{,]/g ; s:/@/g' | xargs npm install --save-dev "$PKG@latest"
)
```

(<https://www.npmjs.com/package/eslint-config-airbnb>)

4. В проекте должен присутствовать файл `.eslintrc`, который содержит ссылку на используемые правила. Правила после установки располагаются в папке `node_modules`. Пример файла `.eslintrc`

```
{
  "extends": "airbnb",
  "rules": {
    "no-console": 0,
  }
}
```

5. Улучшите с помощью линтера код:



и выведите результат выполнения функции в консоль.

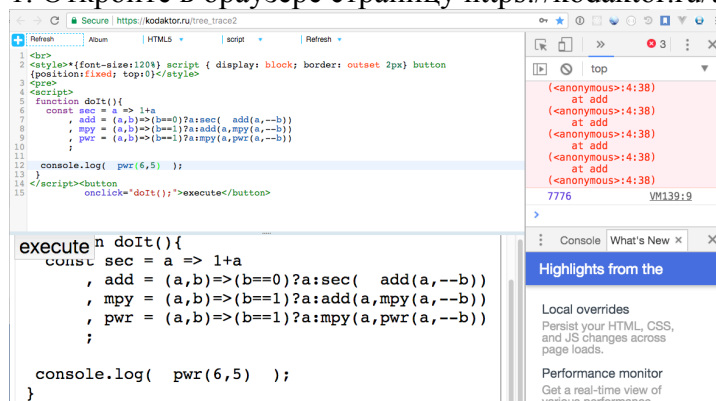
6. Для этого расставьте пробелы и переводы строк согласно рекомендациям линтера и сверьте результат:

```
1 function sayHello(name) {
2   return `Hello, ${name}!`;
3 }
4 console.log(sayHello('Elias'));
5
```

7. Разместите отчёт (лог действий) в репозитории (веб-портфолио).

II. Использование средств разработчика браузера Chrome

1. Откройте в браузере страницу https://kodaktor.ru/tree_trace2



2. Откройте консоль разработчика (⇧ ⌘ J или Ctrl + Shift + J)

3. Выясните пороговые значения сочетания фактических параметров, передаваемых в функцию `pwr`, при которых начинают возникать сообщения об ошибке переполнения стека рекурсивных вызовов

4. Используйте точки останова (breakpoints, <http://kodaktor.ru/recurl.mp4>) чтобы выяснить количества вызовов функций `add`, `mpu` и `pwg`
5. Разместите отчёт (лог действий) в репозитории (веб-портфолио).

2. Типовые задания для инвариантной самостоятельной работы по темам

Задания для самостоятельной инвариантной работы обобщают и расширяют задания лабораторных работ, преимущественно предполагают поисковую деятельность в аспекте модификация и совершенствования алгоритмов и моделей, предложенных в лабораторных работах и лекционном материале по теме. Эти задания нацелены на формирование видов деятельности, которые обучающиеся должны уметь осуществлять. Задания представлены в СДО Moodle в электронном учебном курсе по дисциплине «Веб-проектирование и веб-языки».

Выполнение задания предполагает следующие виды деятельности:

- поиск информации в Интернете по изучаемым технологиям и инструментам, анализ справочных материалов и документации,
- развёртывание и настройку окружения (программной среды, IDE, папки проекта, репозитория);
- проектирование формата данных веб-приложения или формального языка (языка разметки), основанного на XML;
- разработку сценария или приложения на веб-языке (языке программирования высокого уровня в области веб-ресурсов):
 - JavaScript;
 - и, возможно, другом языке по выбору обучающихся;
- выявление и исправление синтаксических ошибок, выполнение транспилиции, тестирования работы сценария.
- составление отчета о выполненном задании в виде слайдов и/или онлайн-документации, размещаемой в Git-репозитори как части веб-портфолио обучающегося.

Критерии оценивания. Задание считается выполненным, если сценарий или отчёт соответствует заданию, сценарий успешно проходит процедуру модульного тестирования и сопровождается репозиторием.

№ темы	Содержание самостоятельной работы обучающихся
1	<p>1. Создание рабочего пространства, регистрация репозитория Git в качестве портфолио, анализ, обоснование выбора и настройка выбранных для работы аппаратно-программных комплексов, современных инструментальных средств удалённого доступа и веб-технологий.</p> <p>2. Проектирование предметного языка формализованного описания или разметки и размещение отчёта по выполнению заданий в веб-портфолио.</p>
2	<p>1. Настройка программного обеспечения для управления зависимостями в веб-проекте</p> <p>2. Проектирование линейного и ветвящегося алгоритма на JavaScript, проектирование рекурсивного алгоритма и фабрики функций на основе</p>

	каррирования и замыкания и размещение отчёта по выполнению заданий в веб-портфолио.
3	<p>1. Разработка приложения для асинхронного считывания данных из JSON и вывода в веб-документ путём нативной модификации дерева DOM</p> <p>2. Проектирование регулярного выражения и сценария валидации веб-формы и размещение отчёта по выполнению заданий в веб-портфолио.</p>
4	<p>1. Настройка линтера и иных средств мониторинга корректности программного кода</p> <p>2. Подготовка наборов модульных тестов и размещение отчёта по выполнению заданий в веб-портфолио.</p>

3. Типовые задания для вариативной самостоятельной работы по темам

Задания для вариативной самостоятельной работы позволяют углубить и детализировать важные аспекты содержания дисциплины, представляющие интерес для каждого конкретного обучающегося, включая углубление и детализацию теоретической подготовки и формирование исследовательской деятельности, для реализации чего предназначены задания на подготовку выступлений, докладов и аналитики.

Для выполнения этих заданий требуется выполнение следующих действий:

- анализ программных средств и инструментов, их документации;
- проектирование стиливых характеристик веб-ресурсов;
- анализ и использование фреймворков, логирование.

При подготовке выступления следует руководствоваться следующей дорожной картой презентации:

- обзор по теме (сравнение, таблица, ... - слайды в google drive или инструменте вещания слайдов);
- демонстрация в live-режиме (slides.com, например <http://slides.com/elizabethanatskaya-1/deck-2#/12> и др.);
- выводы;
- примеры заданий для аудитории на овладение материалом (интерактив);
- поддержка в репозитории (ссылки на слайды / ресурсы / ...).

Критерии оценивания. Задание считается выполненным, если сценарий или отчёт соответствует заданию, сценарий успешно проходит процедуру модульного тестирования и сопровождается репозиторием, выступление логично и последовательно охватывает заявленную тему.

№ темы	Содержание самостоятельной работы обучающихся
1	1. Проектирование стилей оформления веб-документа с помощью препроцессора 2. Экспериментальная проверка корректности документов (валидация) на языках разметки
2	1. Подготовка презентации по идиоматике JavaScript и паттернам проектирования. 2. Подготовка материалов для выступления по функциональному программированию на языке JavaScript
3	1. Проектирование клиентского сценария обслуживания корзины покупателя в Интернет-магазине 2. Проектирование сценария для подбора цветового оформления веб-документа

4	<ol style="list-style-type: none"> 1. Подготовка презентации по инструментам веб-разработчика в составе браузера 2. Подготовка материалов для выступления по настройке линтера кода
---	---