

Construção de Software

Primeiro Semestre / 2018

Universidade Federal de Goiás (UFG) -- Instituto de Informática -- Sistemas de Informação
Período: 14/03/2018 a 26/06/2018 (quartas-feiras)
Carga horária: 0 (aulas teóricas), 64 (horas práticas)
Local: INF-153

Fábio Nogueira de Lucena
fabio@inf.ufg.br

Ementa

Definição de construção de software. Pré-requisitos para a construção. Boas práticas para definição dos requisitos. Arquitetura de software e seus componentes. Escolha da linguagem de programação. Convenções de programação. Principais práticas de construção de software. Projeto de software (conceitos, práticas, níveis e abordagens comuns). Formas de acoplamento. Classes e pacotes. Rotinas (métodos). Projeto de software em nível de rotina. Motivos para se criar uma rotina. Bons nomes para uma rotina. Tamanho adequado de uma rotina. Programação defensiva. Problemas gerais no uso de variáveis. Nomes adequados para variáveis. Tipos de dados fundamentais. Estruturas. Ponteiros e referências. Dados globais. Organizando código linear. Sentenças de decisão. Sentenças de iteração. Pesquisas em tabelas. Expressões lógicas. Blocos. Instruções nulas e aninhamentos profundos. Estruturas de controle e complexidade. Qualidade de software. Construção colaborativa. Testes de desenvolvedor. Depuração. Refatoração. Estratégias e técnicas de otimização de código. Relação entre tamanho do código e construção. Gerenciando a construção. Integração. Ferramentas de programação. Leitura e estilo.

Pré-requisitos

O bom aproveitamento nesta disciplina depende do domínio de tópicos cobertos em outras disciplinas. Aqueles mais próximos são identificados abaixo, juntamente com a expectativa correspondente por disciplina. Não é recomendado fazer essa disciplina sem aproveitamento nas disciplinas abaixo.

- Engenharia de Software. Você possui noção de software e de questões pertinentes ao seu desenvolvimento de maneira ampla.
- Engenharia de Requisitos. Você sabe especificar os requisitos, ler e compreender especificações de requisitos.
- Projeto de Software. Você possui noções de como “organizar” um software para atender os requisitos especificados.
- Banco de Dados 1. Você sabe modelar as informações pertinentes a um problema.
- Lógica. Você possui fundamentos para elaboração de algoritmos.
- Introdução à Programação. Você sabe o que é um programa e como programar (transformar um algoritmo em código), já fez uso de ferramentas de edição de software.
- Programação Imperativa. Você domina elementos da construção de software como ambientes integrado de desenvolvimento, estruturas de dados, recursão e outros.
- Estrutura de Dados. Você sabe organizar dados em uma estrutura adequada para o correspondente processamento.
- Programação Orientada a Objetos. Você possui domínio sobre o paradigma orientado a objetos.
- Programação para Web. Você possui conhecimento e prática na produção de back-end para aplicações web (web apps).

Visão geral da disciplina

A disciplina fará uso da linguagem JavaScript para a realização das atividades práticas. Convém lembrar que esta disciplina é 100% prática. Por meio dos exercícios os tópicos da ementa serão abordados.

Cada aula terá um conjunto de exercícios pertinentes para o qual o docente irá orientar o aspecto de construção a ser observado de forma criteriosa. O docente estará todo o tempo da aula, após a explanação (orientação), disponível para atender os estudantes e auxiliá-los na realização do que é pedido. A realização das atividades é individual e os produtos resultantes serão empregados na determinação da nota final da disciplina.

É imprescindível dedicação desde a primeira aula para um adequado aproveitamento do conteúdo.

IMPORTANTE: além das atividades previstas nas atividades supervisionadas para cada aula, devem ser acrescentados o que for necessário para concluir os objetivos da aula em questão.

Tópico 1 (4) : 14/03/2018, 21/03/2018, 28/03/2018 e 04/04/2018

- Apresentação do plano da disciplina.
- [Github](#) (criar conta, caso não possua). Repositório do estudante e no qual o docente deve ter acesso. Acrescente o docente (conta **kyriosdata**), como colaborador. O repositório será utilizado para registrar as atividades práticas do estudante e deve ter como nome: **cs-2018** (letras minúsculas).
- [Git](#) (sistema de controle de versões).
- Exercício inicial (fornecido pelo docente) ([aqui](#)).
- Apresentação detalhada do algoritmo para resolução do problema.
- Implementação do algoritmo (pelos estudantes usando a linguagem de programação desejada).
- Questionário de avaliação (perfil).

Atividades supervisionadas (1 hora)

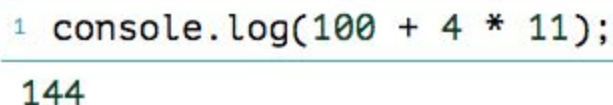
- Ambientação com o git e github.
- Exercício “dia da semana” ([aqui](#)).
- Um possível algoritmo para resolução do exercício ([aqui](#)).
- Uma implementação em JavaScript ([aqui](#)) (em formato e meio inconveniente, mas o empregado no momento).

Tópico 2 (5): 11/04/2018

- Visão de construção de software (e do ambiente de desenvolvimento a ser usado).
- Verifique se possui acesso às ferramentas necessárias (veja [como](#)): (a) git; (b) node; (c) npm e (d) qunit.
- Contexto: JavaScript, navegadores e Node.js.
- Ambientação com JavaScript. [Eloquent JavaScript](#) é grátis e online, com possibilidade de interação com os exemplos. Ao longo do texto você verá trechos de código como

`100 + 4 * 11`

Quando se clica no trecho de código abre-se um editor, com recursos para edição e execução do mesmo (Ctrl-Enter), conforme ilustrado abaixo.



The screenshot shows a code editor with a light blue border. Inside, the first line of code is `1 console.log(100 + 4 * 11);` with a line number '1' on the left. Below the code, the output '144' is displayed.

- O foco da aula é em elementos já conhecidos (seguramente já dominados, mesmo que em outras linguagens): tipos; operadores; formas de representação de strings; números; typeof; NaN (0/0); Infinity (1/0); null; undefined e curto-circuito de operadores lógicos em JavaScript. Minha referência preferida já citei, a segunda é [W3Schools](#).

Atividades supervisionadas (1 hora)

- Ambientação com o git e github.
- Ambientação com JavaScript.
- Na sua conta individual no Github e no seu repositório pessoal (cs-2018), criar o diretório **topico-2** (sem acento) e depositar nele um arquivo correspondente a cada um dos exemplos [aqui](#). Experimente a execução de cada arquivo usando o Node.js.

Tópico 3 (6): 18/04/2018

- Ambientação com a linguagem JavaScript: estrutura de um programa; sentenças e expressões; `alert`; `prompt`; `console.log`; fluxo de controle (`for`, `while`, `break`); estrutura de decisão (`if`, `switch`); comentários; funções; parâmetros; escopo de parâmetros; funções como valores; *arrow functions*; valor padrão para parâmetro omitido; *closure*. Conforme tópico anterior, a sugestão é [Eloquent JavaScript](#) para esta finalidade.
- Ambientação com JavaScript (teste executado em navegador) ([aqui](#)).

Atividades supervisionadas (2 hora)

- Realizar atividade prática (vale nota). No portal <https://coderbyte.com> há uma variante do seguinte problema: criar uma rotina que recebe uma sequência de caracteres, apenas letras [a-z], além de espaço em branco, e produz na saída uma sequência contendo apenas as letras, sem os espaços, mas em ordem alfabética. Por exemplo, se a entrada é **casa** então a saída é **aacs**. Outro exemplo, se a entrada é **a vida e bela**, então a saída é **aaabdeeilv**. Na sua conta individual no Github e no seu repositório pessoal (cs-2018), criar o diretório `topico-3`, no qual deverá ser depositada a implementação desta rotina juntamente com o código que o testa, empregando a ferramenta QUnit. Em tempo, deve ser possível executar o teste via navegador conforme ilustrado [aqui](#).
- Saiba o que é [NodeJS](#) e qual o seu propósito.

Tópico 4 (7): 25/04/2018

- Noções básicas de testes de unidade.
 - Definição ([aqui](#))
 - Como escrever bons testes? ([aqui](#)) ([aqui](#))
 - Ampla compreensão de testes de unidade ([aqui](#))
- Prática com QUnit.

Atividades supervisionadas (1 hora)

- Compare sua resposta com aquela disponível [aqui](#).
- Leia o texto e implemente os algoritmos 1 e 2 (vale nota) ([aqui](#)).

Tópico 5 (10): 02/05/2018, 09/05/2018 e 16/05/2018

- Fluência em JavaScript ([aqui](#)). O que inclui:
 - Criação e uso de módulos.
 - Ambientação com o repositório público NPM (<https://www.npmjs.com/>).
 - Ambientação com classes e métodos (recursos oferecidos) pela Node.js API (<https://nodejs.org/dist/latest-v10.x/docs/api/>).
 - Consulte a [introdução](#) acerca de JSDoc ou inicie por [aqui](#).

Atividades supervisionadas (3 horas)

- Ler, compreender e executar os exemplos de fluência em JavaScript ([aqui](#)).
- Implementar o algoritmo 3 (vale nota) ([aqui](#)). A implementação deve fazer uso de:
 - (a) testes de unidade; (b) documentação realizada em conformidade com JSDOC 3 (<http://usejsdoc.org/>); (c) o algoritmo deverá ser implementado como método de uma classe; (d) a classe será fornecida em um arquivo no qual deve estar presente apenas a classe; (e) o uso da classe deve ocorrer por meio de “import”; (f) classe deverá ser disponibilizada publicamente no repositório NPM; e (g) uma aplicação via linha de comandos que permite fornecer uma data (entrada padrão) e que produz, na saída padrão, o dia da semana correspondente à data. Em tempo, a aplicação a ser criada deve fazer uso da classe disponibilizada publicamente.

Tópico 6 (11): 23/05/2018

- Análise estática de código em JavaScript.
- Siga a apresentação ([aqui](#)) e ambiente-se com as ferramentas e outras fontes citadas, ou seja, saiba pelo menos o que significam, qual o uso pretendido, qual o resultado esperado.
- Exemplo de integração contínua ([aqui](#)).

Atividades supervisionadas (1 hora)

- Implementar o algoritmo 4 (vale nota) ([aqui](#)). Implementação significa fazer uso dos recursos de testes de unidade, documentação, “bom” *design* para o código, registro adequado no Github e, adicionalmente, uso de ferramenta para análise estática. Você deverá selecionar e fazer uso de uma delas. Crie um arquivo **package.json** correspondente (detalhes na documentação do [npm](#)).

Tópico 7 (12): 30/05/2018

- NOTA DA REITORIA (pertinente à greve dos caminhoneiros). Em decorrência desta nota, nossa aula desta quarta-feira será NÃO PRESENCIAL.
- Detector de número mágico em JavaScript.
- Se a ferramenta [ESLint](https://eslint.org/docs/rules/no-magic-numbers) é empregada, então você pode tratar a ocorrência de números mágicos como erro (<https://eslint.org/docs/rules/no-magic-numbers>).
- Objetivo: reforçar o uso de analisadores estáticos (iniciado desde o tópico anterior).

Atividades supervisionadas (1 hora)

- Implementar o algoritmo 5 (vale nota) ([aqui](#)). As observações feitas para o tópico anterior vale para este e para as demais atividades (apenas reiterando para que não persista nenhuma dúvida). Além dos elementos anteriores, você deverá configurar ou usar uma ferramenta que considere a presença de números mágicos. Para verificar que, de fato a ferramenta está fazendo a verificação estática, introduza um número mágico para para que o “erro” seja detectado pela ferramenta empregada.

Tópico 8 (junho/2018): 06, 13, 20 e 27 (quatro últimos encontros)

- O tópico 8 reúne os quatro encontros finais.
- Trata-se do momento de reutilizar todos os recursos empregados até o momento: (a) documentação do código; (b) testes de unidade; (c) analisadores estáticos para a produção de código “com qualidade” em JavaScript e (d) disponibilizar código no repositório NPM.

Atividades supervisionadas (1 hora)

- Implementar os algoritmos 6, 7, ..., 21 (vale nota) ([aqui](#)). Ao todo são 16 algoritmos. O resultado será uma biblioteca contendo as implementações de todos os algoritmos. Esta biblioteca, uma única biblioteca, deverá ser disponibilizada no NPM.
- Leitura sobre integração contínua para JavaScript ([aqui](#)).