

APRENDIZAGEM EM FOCO

---

# FRAMEWORKS FRONT END



# APRESENTAÇÃO DA DISCIPLINA

Autoria: Anderson da Silva Marcolino

Leitura crítica: Paulo Henrique Santini

O desenvolvimento de aplicações e websites, na maioria das vezes, é algo complexo, que demanda esforços de diversos profissionais para agradar aos usuários finais e também aos solicitantes de tais serviços.

Neste contexto, é necessário que as equipes estejam alinhadas com tecnologias recentes, além de bem estruturadas, para que garantam produtos de qualidade. Destarte, a utilização de recursos que facilitem o processo de desenvolvimento e suas várias etapas, bem como reduzam o tempo de entrega das demandas, levam à utilização de *frameworks front end*.

Considerando as perspectivas mais atuais de mercado, tais *frameworks*, ou arcabouços, em português, adicionam qualidade as soluções, já que os desenvolvedores devem utilizar seus recursos sintáticos e semânticos para isso, bem como permitirem melhor alinhamento e padronização do projeto final, sem contar a vasta gama de recursos que podem ser reutilizados, garantindo a entrega das demandas para ambientes de produção, de modo mais rápido.

É com base nestes aspectos da utilização de *frameworks* para as interfaces e soluções web ou *front end*, que esta disciplina tem como objetivo permitir conhecer e identificar as vantagens e desvantagens dos *frameworks*, suas especificidades em termos sintáticos e semânticos, bem como integrá-los a projetos existentes e utilizá-los em conjunto com sistemas de bancos de dados e interfaces de programação de aplicações (API).

Com os fundamentos e conhecimentos destas tecnologias, você poderá criar websites e aplicações web mais facilmente e estará em sintonia com o que há de mais moderno em termos de desenvolvimento de interfaces web! Bons estudos!

## INTRODUÇÃO

Olá, aluno (a)! A *Aprendizagem em Foco* visa destacar, de maneira direta e assertiva, os principais conceitos inerentes à temática abordada na disciplina. Além disso, também pretende provocar reflexões que estimulem a aplicação da teoria na prática profissional. Vem conosco!



**TEMA 1**

# Diferenciando e escolhendo um Framework Front End

---

Autoria: Anderson da Silva Marcolino

Leitura crítica: Paulo Henrique Santini



## DIRETO AO PONTO

Para o desenvolvimento de aplicações web robustas, as empresas de desenvolvimento web acabam adotando diferentes tecnologias e ferramentas que possam trazer benefícios relacionados, principalmente, à redução de custos, otimização e mais agilidade na entrega dos produtos e, por último, mas não menos importante, a facilidade de manutenção e evolução da solução de software.

Nesta perspectiva, a adoção de *frameworks* que possam auxiliar na obtenção de tais benefícios é primordial. Logo, notamos diversas opções que podem ser adotadas. Considerando a linguagem de programação JavaScript e outras linguagens derivadas da mesma, como TypeScript, podemos destacar três *frameworks*: Vue.js, Angular e React.

O Quadro 1 apresenta um resumo das principais diferenças entre eles, o que pode nos apoiar a selecionar um dos *frameworks*, além de entender sobre o quanto fácil é obter suporte para documentações e materiais que possam auxiliar na sua utilização e também na solução de possíveis problemas, ao ser adotado no desenvolvimento de aplicações web.

Característica	Vue.js	Angular	React
Mantenedor.	Comunidade independente.	Google	Facebook.
Framework/ biblioteca se integra com outros frameworks.	Sim.	Sim – porém é mais restrito.	Sim.
Características específicas.	Possibilidade com integração em elemento no HTML e também por meio do Node.js.	Utilização somente via Node.js.	Possibilidade com integração em elemento no HTML e também por meio do Node.js.
Tipo de padrão. Arquitetural indicado.	Inspirada na Visão-Modelo do Modelo-Visão-Visão-Modelo.	Sem padrão específico.	Inspirada na Visão do Modelo-Visão-Controle.

Documentação.	Possui documentação de acesso fácil.	Possui documentação de acesso fácil.	Possui documentação de acesso fácil.
Comunidade.	Possui GitHub, fóruns e ambientes para tirar dúvidas.	Possui GitHub e ambientes para tirar dúvidas. Não há fórum oficial.	Possui GitHub, fóruns e ambientes para tirar dúvidas.
Tamanhos dos pacotes de arquivos.	Pequenos.	Médios.	Pequenos.
Tempo de Renderização	Rápida renderização.	Renderização mais lenta.	Rápida renderização.
Nível de dificuldade (de 0 a 10 – Quanto maior, mais difícil de se aprender).	9.	8.	7.
Nota (de 0 a 10).	8.	8.	9.

Fonte: adaptado de Ferreira (2018); De Camargos (2019), Guia (2021); Angular(2021) e React (2021).

Desse modo, podemos notar que, apesar das diferenças, são poucos pontos negativos para os três *frameworks* mencionados. Como veredito final para a escolha do melhor, notamos que React e Vue são mais robustos. Contudo, as vantagens são pequenas e o que importa é a aptidão da equipe de desenvolvimento e usar ou não uma das opções.

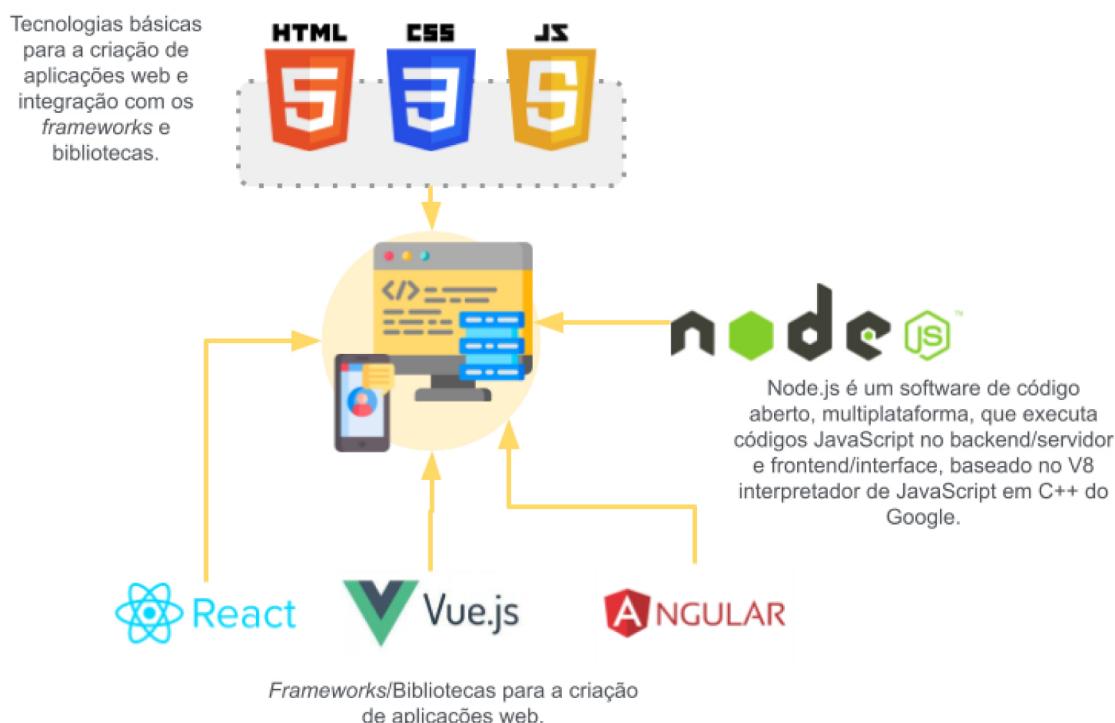
O infográfico da Figura 1 apresenta a integração das tecnologias utilizadas para a criação das aplicações web.

Podemos conceber tais aplicações utilizando apenas linguagem de marcação de hipertexto (HTML), folha de estilos em cascata (CSS) e JavaScript. Contudo, os benefícios em utilizar *frameworks* ou bibliotecas que trazer muitas funções já prontas para utilização são preferíveis do que criar tudo do zero. Nesse sentido, tais tecnologias ainda são essenciais, pois são integradas aos *frameworks* e bibliotecas adotadas, além de fornecerem conhecimento básico para a utilização de algo mais avançado.

Em relação ao Node.js, que é um construtor de aplicações web JavaScript, todas as indicações de *frameworks* e bibliotecas podem

fazer uso do mesmo. Angular, por exemplo, obrigatoriamente usa o software para compilar as aplicações. Já React e Vue.js podem ser utilizados diretamente nas páginas HTML, além de poderem ser utilizadas com o Node.js. A adoção ou não do Node dependerá, principalmente, da complexidade e tamanho do projeto.

**Figura 1 – Infográfico da relação entre as tecnologias**



Fonte: elaborada pelo autor.

## Referências bibliográficas

- FERREIRA, H. K.; ZUCHI, J. D. Análise comparativa entre *frameworks frontend* baseados em JavaScript para aplicações web. Revista Interface Tecnológica, v. 15, n. 2, p. 111-123. Taquaritinga, 2018.
- GUIA VUE.JS. Documentação Oficial. 2021.
- ANGULAR. Documentação Oficial. 2021.
- REACT. Documentação Oficial. 2021.

DE CAMARGOS, J. G. C. et al. Uma análise comparativa entre os frameworks Javascript Angular e React.: **Computação & Sociedade**, v. 1, n. 1. Belo Horizonte, 2019.

## PARA SABER MAIS

Você sabe o que é o Node.js e como pode facilitar o uso de *frameworks* ou bibliotecas, como o Vue.js, Angular e React?

Node.js foi projetado para construir aplicações para redes escaláveis, sendo um software para execução de código JavaScript em tempo de execução, dirigido a eventos. Pode ser utilizado para dar suporte na criação de projetos, tanto *front-end* quanto *back-end* (NODE.JS, 2021).

Considerando os **frameworks** e bibliotecas Vue, Angular e React o Node.js pode auxiliar na criação, desenvolvimento, testes, manutenção e construção para produção (*deploy*) de modo facilitado.

No contexto da criação, ao instalar o node.js, podemos fazer uso de recursos e comandos que estão integrados ao *Node Package Manager*, que permite instalar bibliotecas, *frameworks* ou executar de modo a criarem toda a estrutura necessária para a criação do projeto base.

Em relação ao desenvolvimento, a integração facilitada de outras bibliotecas auxilia na verificação, validação e testes dos códigos e, no geral, na manutenção do projeto.

Finalmente, integra meios de permitir a criação (*build*) da aplicação para produção, minificando scripts, ou seja, otimizando-os para serem utilizados na Internet, além de verificar possíveis erros que possam impedir o funcionamento correto da aplicação.

Desse modo, entender e usar os recursos do Node.js podem ser fundamentais para garantir melhor desenvolvimento e melhor qualidade na aplicação web final, mesmo utilizando *frameworks* ou bibliotecas robustas. Logo, recomenda-se seu estudo.

## Referências bibliográficas

NODE.JS. Documentação Oficial. 2021.

### TEORIA EM PRÁTICA

O desenvolvimento de aplicações web é complexo e requer esforços da equipe de desenvolvimento, uma vez que deverão, com base na solicitação dos clientes, selecionar um conjunto de tecnologias que possam ser utilizadas de modo adequado e que atendam aos requisitos dos usuários. Nessa perspectiva, indique duas características dos *frameworks* e bibliotecas apresentadas que você apontaria como essenciais para justificar a seleção de um *framework* em relação ao outro. Justifique sua resposta com exemplos fundamentados na teoria deste tema.

**Para conhecer a resolução comentada proposta pelo professor, acesse a videoaula deste *Teoria em Prática* no ambiente de aprendizagem.**

### LEITURA FUNDAMENTAL

#### Indicações de leitura

**Prezado aluno, as indicações a seguir podem estar disponíveis em algum dos parceiros da nossa Biblioteca Virtual (faça o log**

**in por meio do seu AVA), e outras podem estar disponíveis em sites acadêmicos (como o SciELO), repositórios de instituições públicas, órgãos públicos, anais de eventos científicos ou periódicos científicos, todos acessíveis pela internet.**

**Isso não significa que o protagonismo da sua jornada de autodesenvolvimento deva mudar de foco. Reconhecemos que você é a autoridade máxima da sua própria vida e deve, portanto, assumir uma postura autônoma nos estudos e na construção da sua carreira profissional.**

**Por isso, nós o convidamos a explorar todas as possibilidades da nossa Biblioteca Virtual e além! Sucesso!**

## **Indicação 1**

Este livro aponta as principais técnicas para desenvolver uma página ou aplicação web acessível. Visto que conteúdos web precisam ser adaptados para pessoas com necessidades especiais, além de serem atrativas para os demais usuários, entender e criar meios de se desenvolver para web garantindo a acessibilidade gera um diferencial significativo no intuito de desenvolver, principalmente, considerando *frameworks* e bibliotecas web.

Para realizar a leitura, acesse a plataforma Biblioteca Virtual e busque pelo título da obra no parceiro *Biblioteca Senac*.

**FERRAZ, R. Acessibilidade na web.** São Paulo: Senac, 2017.

## **Indicação 2**

Esta apostila permite a revisão dos conceitos essenciais e avançados sobre HTML, CSS e JavaScript, fundamentais para a

posterior migração para um *framework* ou biblioteca JavaScript. Adicionalmente, são conteúdos base para o desenvolvimento de aplicações e páginas web.

Para realizar a leitura, acesse a plataforma Biblioteca Virtual e busque pelo título da obra no campo de busca por Título.

**K19. Desenvolvimento Web com HTML, CSS e Javascript.** São Paulo: K19 Treinamentos, 2015.

## QUIZ

**Prezado aluno, as questões do Quiz têm como propósito a verificação de leitura dos itens *Direto ao Ponto, Para Saber Mais, Teoria em Prática e Leitura Fundamental*, presentes neste Aprendizagem em Foco.**

**Para as avaliações virtuais e presenciais, as questões serão elaboradas a partir de todos os itens do Aprendizagem em Foco e dos slides usados para a gravação das videoaulas, além de questões de interpretação com embasamento no cabeçalho da questão.**

1. Os *frameworks* e bibliotecas auxiliam significativamente no desenvolvimento web. Considerando as características que levam o Vue.js, Angular e React a serem adotadas, assinale a alternativa correta:
  - a. Angular não possui suporte à integração com outros *frameworks* e bibliotecas.
  - b. React não disponibiliza repositório Git com o código de sua biblioteca.

- c. Vue.js necessita da integração com o Node.js para ser adotado em projetos para o desenvolvimento de aplicações web.
  - d. Angular não possui a especificação de um padrão arquitetural que se baseia.
  - e. React possui tempo de renderização mais lento, bem como apresenta os tamanhos de pacotes médios.
2. Os *frameworks* e bibliotecas trazem diversos benefícios ao processo de desenvolvimento web. Considerando, de modo geral, os *frameworks* e bibliotecas como Vue.js, Angular e React, e suas vantagens no desenvolvimento de aplicações web, assinale a alternativa correta:
- a. *Frameworks* e bibliotecas auxiliam na redução de membros em uma equipe de desenvolvimento.
  - b. *Frameworks* e bibliotecas reduzem o custo especificamente na etapa de testes das aplicações.
  - c. *Frameworks* e bibliotecas agilizam o processo de adoção de uma nova tecnologia.
  - d. *Frameworks* e bibliotecas melhoram a qualidade nos produtos e otimiza a entrega de aplicações web.
  - e. *Frameworks* e bibliotecas devem ser escolhidos sempre pela quantidade de desenvolvedores que os conhecem.



## GABARITO

### Questão 1 - Resposta D

**Resolução:** Tanto o Vue.js, quanto o Angular e React possuem suporte à integração com outros *frameworks* e bibliotecas, a grande diferença é que este suporte é mais restrito em relação ao Angular. Todos os *frameworks* e bibliotecas possuem

repositório Git. Já em relação a integração obrigatória com o Node.js, apenas o Angular o exige. Adicionalmente, o Angular não possuir uma especificação definida do padrão arquitetural que atende, o que não exime de sua potencialidade no uso de projetos. Contudo, é o *framework* que demanda mais tempo de renderização e pacotes de tamanho médio, quando comparados ao React e o Vue.js, que são mais rápidos e consomem menos espaço de armazenamento.

## Questão 2 - Resposta D

**Resolução:** Os *Frameworks* e bibliotecas trazem diversos benefícios ao processo de desenvolvimento. Mas podem também trazer complicações ou não interferir em determinados elementos envolvidos no desenvolvimento. Não é possível afirmar que os membros de uma equipe serão reduzidos somente pelo fato de se adotar um *framework* ou biblioteca. Pode ser que a demanda por utilização de uma nova tecnologia até aumente a equipe. O mesmo ocorre ao se afirmar que reduz o custo, majoritariamente, na etapa de testes. Geralmente, *frameworks* e bibliotecas auxiliam, mas as várias etapas de desenvolvimento e não somente a de testes. É uma falácia dizer que adotando um *framework* ou biblioteca, temos a agilização no processo de adoção de tecnologias. Geralmente, ocorre o contrário, visto que membros da equipe podem necessitar estudar o *framework* para, então, permitir seu uso. Entretanto, é verdade que os *frameworks* e bibliotecas buscam melhorar a qualidade dos produtos e otimizar a entrega dos mesmos. Finalmente, a escolha de um *framework* ou biblioteca em relação a outro não deve ocorrer por conveniência dos desenvolvedores que os conhecem, mas pela solução que busca alcançar por determinadas tecnologias.



**TEMA 2**

## **Compreendendo e utilizando o Framework Vue.js**

---

Autoria: Anderson da Silva Marcolino

Leitura crítica: Paulo Henrique Santini



## DIRETO AO PONTO

O Vue.js é um *framework* JavaScript e, considerando suas vantagens, podemos indicar sua capacidade de reuso por meio da criação de componentes. Um componente é formado por alguns elementos essenciais, que estão resumidos no Quadro 1.

**Quadro 1 – Elementos para Criação de Componentes com Vue.js**

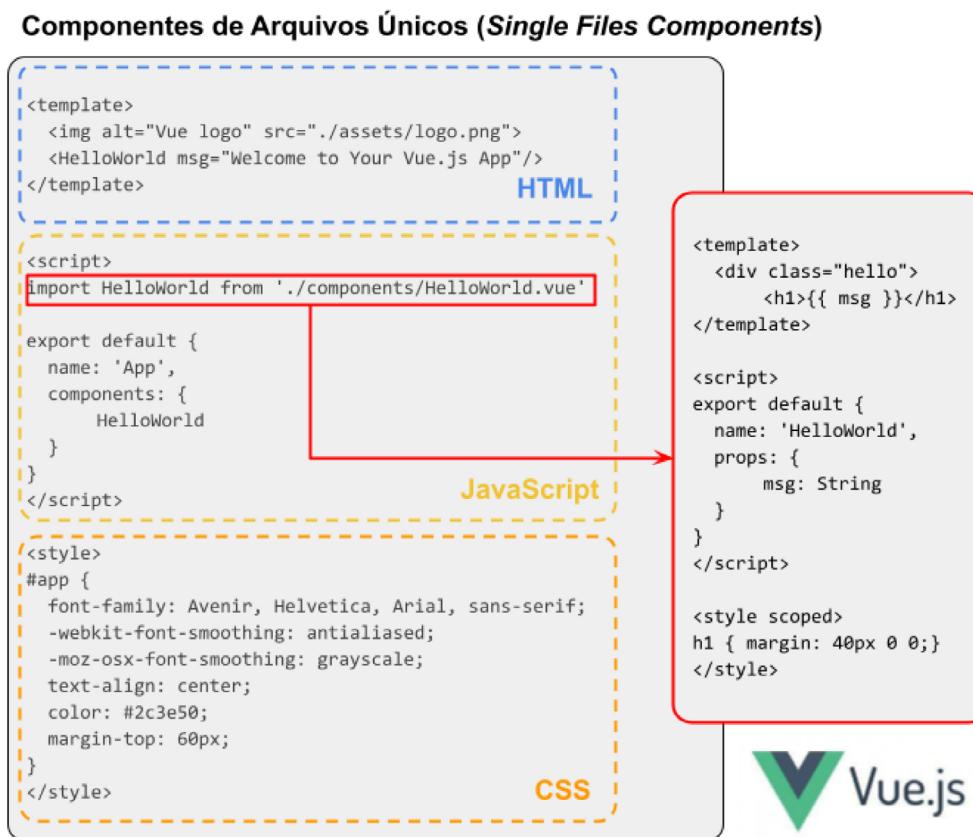
Arquivo	Código	Descrição
Index.html	<div id="app"></div>	Elemento DOM no <i>document</i> HTML, que renderizará a aplicação Vue.js. Geralmente, os projetos possuem apenas uma instância <i>app</i> na aplicação toda.
src/main.js	new Vue({ el: '#app' })	Instância, uma aplicação vue.js. Na propriedade <i>el</i> deve-se indicar o <i>selector</i> que será utilizado no elemento DOM para garantir a renderização da aplicação.
src/components/mensagem.vue	Vue.component('nome-usuario', { props: ['nome'], template: '<p>Olá {{ nome }}</p>' })	Cria o componente, primeiro indicando seu nome de marcação ( <i>nome-usuario</i> ), as propriedades ( <i>props</i> ), que poderão ser passadas do elemento DOM para o componente, e a propriedade de <i>template</i> , que representa o que de fato será exibido para o usuário, que exibe, por sua vez o conteúdo da <i>props nome</i> .

src/App.vue	<pre>const Sidebar = {   template:     '&lt;aside&gt;Sidebar&lt;/     aside&gt;'   }   new Vue({     el: '#app',     components: {       Sidebar     }   }).</pre>	Componente declarado em uma variável que pode ser utilizada apenas localmente, devido seu encapsulamento. O componente é vinculado a aplicação, por meio da propriedade <i>components</i> na instância da aplicação ( <i>new Vue({...})</i> ).
-------------	--	--

Fonte: adaptado e traduzido de Guide (2021).

A Figura 1, abaixo, apresenta dois componentes de arquivo único. Tais componentes concentram, em seu conteúdo, um template que corresponde à parte do código em HTML, o *script* que corresponde a parte em JavaScript e o *style*, que corresponde à folha de estilo em cascata. Um componente de arquivo único tem a vantagem de concentrar tudo o que é necessário para sua apresentação e integração à outros componentes. No exemplo, o componente *HelloWorld* apresentado no quadro, em coloração vermelha, é importado e utilizado em outro componente, que recebe como propriedades um texto que será exibido por meio da propriedade de nome *msg*.

## Figura 1 – Especificação de dois componentes de arquivos únicos



Fonte: elaborada pelo autor.

## Referências bibliográficas

GUIDE. **Vue.js Documentation**. 2021.



Você sabia que o vue.js utiliza, em sua marcação de template, um conjunto de elementos do HTML? Você sabe qual o papel de um template nos projetos do vue?

Vue.js utiliza uma linguagem para a criação de templates, que é um conjunto de elementos preexistentes no HTML. Tal utilização garante dois recursos importantes para uma aplicação desenvolvida, considerando tal *framework*: a interpolação e diretivas.

Um template vue.js é exemplificado a seguir:

```
<span>Olá!</span>
```

Este mesmo template pode ser declarado em um componente vue:

```
new Vue({  
  template: '<span>Olá!</span>'  
})
```

Ainda, pode ser inserido também em um componente de arquivo único:

```
<template>  
<span>Olá!</span>  
</template>
```

Este exemplo básico mostra apenas como um template pode ser utilizado no contexto de um projeto vue. Considerando a vantagem do uso de templates, o código a seguir apresenta a interpolação.

```
new Vue({  
  data: {  
    nome: 'Daniela'  
  }  
})
```

```
},  
  
template: '<span>Olá {{nome}}!</span>'  
})
```

Note que a informação nome é usada e inserida no template por meio da sintaxe das duas chaves ("{{}}"). Não foi necessário utilizar o acesso ao dado, como seria realizado se considerarmos o JavaScript: *this.data.nome*. Isso ocorre porque o Vue realizar uma ligação (*binding*) que permite que o template possa utilizar a propriedade.

É possível ainda utilizar funções dentro de um bloco de interpolação:

```
{{ nome.reverse() }}
```

Entretanto, este uso é limitado a uma única expressão por bloco de interpolação. É recomendado não utilizar lógicas complexas em *templates* Vue, mas a possibilidade de uso, ainda que restrito, traz boa flexibilidade, uma vez que os valores de dados exibidos, se alterados na aplicação, refletem em atualização imediata do conteúdo na interface.

Para evitar o uso de interpolação por meio das chaves duplas, pode-se utilizar uma diretiva no elemento do *template*, cujo o conteúdo de uma propriedade pode ser exibido. Desse modo, o código anterior seria:

```
new Vue({  
  
  data: {  
  
    nome: 'Daniela'  
  
  },
```

```
template: 'Olá <span v-text= "nome"></span>!'  
})
```

No caso, a diretiva *v-text* insere o conteúdo da propriedade nome no local de conteúdo do elemento do template *span*. Outro exemplo de uso de diretiva é:

```
<span v-text="nome == Maria ? 'Olá Maria!' : 'Olá' + nome  
+ '!'"></span>
```

Assim, Vue possui uma série de diretivas que podem ser utilizadas para representar elementos nos templates. Recomenda-se consultar e utilizar a documentação oficial para que possam ser ainda mais exploradas em suas aplicações (GUIDE, 2021).

## Referências bibliográficas

GUIDE. **Vue.js Documentation**. 2021.

## TEORIA EM PRÁTICA

A interpolação de valores entre funções e ações programadas em JavaScript e as interfaces em HTML requerem atenção dos desenvolvedores na hora de garantir que informações sejam refletidas igualmente, para ambas as tecnologias: interface e código JavaScript. Considerando que o desenvolvimento puramente em JavaScript requer maiores esforços para a integração do que as providas pelo *framework* Vue.js, exemplifique, por meio de um projeto em Vue.js, ou seja, por meio de um exemplo prático, como a interpolação, considerando o uso da diretiva *v-text*, ocorre. Utilize a mesma para apresentar informações e descrever seu funcionamento, de fato, em uma aplicação vue.

**Para conhecer a resolução comentada proposta pelo professor, acesse a videoaula deste *Teoria em Prática* no ambiente de aprendizagem.**

## LEITURA FUNDAMENTAL

### Indicações de leitura

**Prezado aluno, as indicações a seguir podem estar disponíveis em algum dos parceiros da nossa Biblioteca Virtual (faça o log in por meio do seu AVA), e outras podem estar disponíveis em sites acadêmicos (como o SciELO), repositórios de instituições públicas, órgãos públicos, anais de eventos científicos ou periódicos científicos, todos acessíveis pela internet.**

**Isso não significa que o protagonismo da sua jornada de autodesenvolvimento deva mudar de foco. Reconhecemos que você é a autoridade máxima da sua própria vida e deve, portanto, assumir uma postura autônoma nos estudos e na construção da sua carreira profissional.**

**Por isso, nós o convidamos a explorar todas as possibilidades da nossa Biblioteca Virtual e além! Sucesso!**

#### **Indicação 1**

Este livro aborda os principais assuntos relacionados à elaboração de um projeto, da concepção aos processos de estruturação e modelagem, incluindo técnicas de gestão. Estuda modelagens essencial e de casos de uso. Explica linguagem de programação, tipo de servidor e banco de dados, bem como a escolha do provedor de hospedagem adequado.

Para realizar a leitura, acesse a plataforma Biblioteca Virtual e busque pelo título da obra no parceiro *Minha Biblioteca*.

ALVES, W. P. **Projetos de sistemas web:** conceitos, estruturas, criação de banco de dados e ferramentas de desenvolvimento. São Paulo: Saraiva Educação S. A., 2019.

## Indicação 2

O livro permite o acompanhamento evolutivo do webdesign, perpassando tópicos como os tipos de sites, seus componentes e as etapas para sua produção (da codificação à manutenção), segundo aspectos como acessibilidade e responsividade. Ademais, examinaremos os padrões técnicos que norteiam esse processo, as ferramentas à disposição do webdesigner e as tendências atuais da área.

Para realizar a leitura, acesse a plataforma Biblioteca Virtual e busque pelo título da obra no parceiro Pearson/Biblioteca Virtual 3.0.

PAZ, M. **Webdesign.** São Paulo: Itersaber, 2021.

## QUIZ

**Prezado aluno, as questões do Quiz têm como propósito a verificação de leitura dos itens *Direto ao Ponto, Para Saber Mais, Teoria em Prática e Leitura Fundamental*, presentes neste Aprendizagem em Foco.**

**Para as avaliações virtuais e presenciais, as questões serão elaboradas a partir de todos os itens do Aprendizagem em Foco**

**e dos slides usados para a gravação das videoaulas, além de questões de interpretação com embasamento no cabeçalho da questão.**

1. O framework Vue.js permite a criação de componentes que, por sua vez, requerem uma série de especificações para serem utilizados. Considerando os componentes vue e sua criação, assinale a alternativa correta:
  - a. No arquivo index.html é definida a chamada principal do script, que inicia a aplicação do vue.
  - b. No arquivo main.js há a instanciação da aplicação vue.js, que, obrigatoriamente, deve possuir a propriedade *el*, que indica o seletor do elemento HTML utilizado para injetar tal aplicação.
  - c. Os componentes que são renderizados na aplicação vue, são criados em arquivos separados, devendo receber um nome igual para todos, de modo que permita sua reutilização.
  - d. Caso um componente não receba um nome de marcação, poderá ser encapsulado e inserido de modo diferenciado na aplicação.
  - e. Para que um componente encapsulado possa ser utilizado, deve ser declarado em um atributo chamado *template* no arquivo App.vue.
  
2. O vue.js possibilita a criação de componentes do tipo arquivo único. Considerando as especificidades de um componente de arquivo único, assinale a alternativa correta:
  - a. Um componente de arquivo único integra folha de estilos e HTML para criar um componente vue.js.
  - b. Um componente de arquivo único integra folha de estilos e JavaScript para criar um componente vue.js.

- c. Um componente de arquivo único integra folha de estilos, o template baseado em HTML e JavaScript para criar um componente vue.js.
- d. Um componente de arquivo único integra JavaScript e o template baseado em HTML.
- e. Um componente de arquivo único integra JavaScript, HTML e PHP para criar um componente vue.js.



## GABARITO

### Questão 1 - Resposta B

**Resolução:** No arquivo index.html, não é definida a chamada principal do script que inicia a aplicação do vue. Neste arquivo, é necessária a definição de um atributo que permita que a inicialização da aplicação do vue possa ser injetada, que, por sua vez, é definido no arquivo main.js. Os componentes dos vue.js devem possuir nomes diferentes para que possam ser reutilizados. Se um componente não receber um nome, não poderá ser inserido na aplicação, mesmo que o objetivo seja seu uso de modo encapsulado. Finalmente, para que um componente possa ser encapsulado e utilizado, deve ser declarado na propriedade *components*, no arquivo App.vue.

### Questão 2 - Resposta C

**Resolução:** Um componente de arquivo único integra folha de estilos, o template baseado em HTML e JavaScript para criar um componente vue.js.



## TEMA 3

# Angular: utilizando o TypeScript com o Framework da Google

---

Autoria: Anderson da Silva Marcolino

Leitura crítica: Paulo Henrique Santini



## DIRETO AO PONTO

O Angular é a versão remodelada do antigo *framework* Angular JS, criado em 2008, na Google. Utiliza a linguagem TypeScript, que é um superconjunto de JavaScript. Desse modo, pode-se utilizar tudo que está disponível na linguagem JavaScript mais o que é disponibilizado em TypeScript.

Angular utiliza componentes para criar aplicações, e esses componentes são compostos por três arquivos principais: um template (HTML), a estilização (CSS) e um arquivo TypeScript, que contém, além da integração dos três arquivos em um *decorator* @ *Component*, a classe principal do componente, onde o código em TypeScript e JavaScript será inserido, e o *selector*, que define o nome do componente para ser utilizado na aplicação.

Ao dividirmos um componente Angular por meio do padrão Modelo – Visão – Controle (do inglês, *model-view-controller*), temos o modelo como sendo arquivo *.ts* e a visão, criada pela união do HTML com CSS. O controle pode ser implementado parcialmente com o modelo, mas é comumente desenvolvido no *back end*, ou seja, na parte da aplicação que é executada no servidor.

Dentre os principais conceitos adicionados pelo TypeScript, temos a inclusão de elementos que permitem que o código seja compilado. O que não ocorre com o JavaScript, por ser uma linguagem interpretada. Logo, é possível identificar erros no código que, ao ser considerado apenas o JavaScript, só eram possíveis identificar ao executar a aplicação.

Dentre as funcionalidades trazidas pelo TypeScript, podemos destacar:

1. Especificação de tipos para os argumentos, propriedades e retornos dos métodos por meio de *type annotations* (anotações do tipo):

```
export class Nome {  
  
    first: string;  
  
    second: string;  
  
    constructor(first: string, second: string) {  
  
        this.first = first;  
  
        this.second = second;  
  
    }  
  
    getMensagem(): string {  
  
        return `Olá ${this.first} ${this.second}`;  
  
    }  
  
}
```

No código, notamos que são especificados os tipos de dados que podem ser atribuídos em variáveis por meio da indicação dos dois pontos e o tipo de dado, por exemplo, *first : string*, que indica o tipo de dado esperado na variável. O mesmo ocorre com os parâmetros passados para o método construtor. Com essa indicação, garante-se que os dados enviados deverão ser do tipo esperado, reduzindo possíveis comportamentos ou erros, no momento da execução da aplicação.

2. Atribuição de vários tipos para uma determinada variáveis, parâmetros e retornos dos métodos:

```
export class ConverteTemperatura {  
  
    static convertFtoC(temp: number | string): string {  
  
        let value: number = <number>temp.toPrecision  
  
        ? <number>temp : parseFloat(<string>temp);  
  
        return ((parseFloat(value.toPrecision(2)) - 32) /  
            1.8).toFixed(1);  
  
    }  
  
}
```

Os múltiplos tipos são atribuídos por meio de uma lista, cujos tipos são separados pelo caractere `|`, chamado de *pipe*. No código anterior, a temperatura passada como parâmetro no método `convertFtoC` pode aceitar, por meio da indicação `: number | string` ambos os tipos de valores. Isso é chamado de união de tipos. Note que ainda é necessário usar o tipo como uma marcação antes da variável `temp`, no corpo do método: `<number>temp` ou `<string>temp`, para o tipo respectivo a ser utilizado. Essa marcação especial garante que a conversão do valor para o tipo marcado, garantindo a execução correta do código, sem possíveis erros.

Uma alternativa par ao uso da união de tipos, é o uso da palavra `any` (do inglês, qualquer um), que permite que qualquer tipo seja atribuído a uma variável, argumento ou retorno. Entretanto, este uso também exige que o restante do código esteja preparado para utilizar a variável de acordo com o tipo que é realmente esperado:

```
export class ConverteTemperatura {  
  
    static convertFtoC(temp: any): string {  
  
        let value: number;  
  
        if ((temp as number).toPrecision) {  
  
            value = temp;  
  
        } else if ((temp as string).indexOf) {  
  
            value = parseFloat(<string>temp);  
  
        } else {  
  
            value = 0;  
  
        }  
  
        return ((parseFloat(value.toPrecision(2)) - 32) /  
            1.8).toFixed(1);  
  
    }  
  
}
```

Note que há uma condicional *if*, declarada como em JavaScript, que verifica se o parâmetro passado é um *number* ou uma *string*: “*temp as number*” ou “*temp as string*”, respectivamente.

### 3. Declaração de modificadores de acesso (encapsulamento):

TypeScript disponibiliza três palavras que são adotadas para indicar as restrições de acesso (encapsulamento), utilizados pelo compilador para verificar se estão sendo atendidas ou não no código. São estes:

- *public*: palavra utilizada para especificar que um método pode ser acessado em qualquer lugar do projeto. É o tipo de visibilidade padrão, caso nenhuma palavra que limite o acesso seja utilizada.
- *private*: palavra utilizada para especificar que a propriedade ou método assim marcado, poderá ser utilizado apenas pela classe que o define.
- *protected*: palavra utilizada para indicar que o método ou propriedades podem ser acessado apenas pelas classes que os definem ou pelas classes que estendem a classe que os definiu.

O código a seguir exemplifica a utilização de modificadores de acesso:

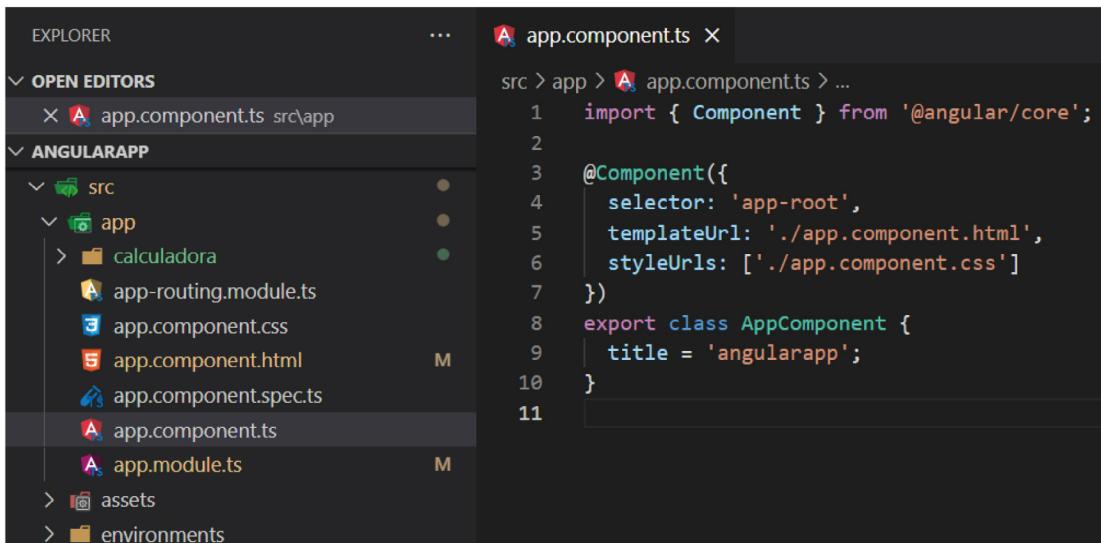
```
export class ConverteTemperatura {  
  
    static convertFtoC(temp: any): string {  
  
        let value: number;  
  
        if ((temp as number).toPrecision) {  
  
            value = temp;  
  
        } else if ((temp as string).indexOf) {  
  
            value = parseFloat(<string>temp);  
  
        } else {  
  
            value = 0;  
  
        }  
    }  
}
```

```
        return ConverteTemperatura.executarCalculo(value).  
       toFixed(1);  
  
    }  
  
    private static executarCalculo (value: number):  
    number {  
  
        return (parseFloat(value.toPrecision(2)) - 32) /  
        1.8;  
  
    }  
  
}
```

Note que o método `executarCalculo` está marcado como privado, o que resultará em erro pelo compilador do TypeScript, caso qualquer outra parte da aplicação tente invocar tal método.

A Figura 1 apresenta a estrutura de arquivos de um projeto Angular e o componente principal `app.component.ts`. Neste arquivo de extensão “.ts” podemos visualizar o *decorator* `@ Component`. É neste *decorator* que temos a indicação do *template* (arquivo HTML), da folha de estilos (arquivo CSS) e o *selector*, que indica a *tag* que deverá ser inserida no `app.component.html` para permitir sua renderização no navegador.

**Figura 1 – Estrutura e componente TypeScript**



The screenshot shows the Visual Studio Code interface. On the left is the Explorer sidebar, which lists the project structure under 'ANGULARAPP'. It includes 'src' (with 'calculadora'), 'app' (with 'app-routing.module.ts', 'app.component.css', 'app.component.html', 'app.component.spec.ts', 'app.component.ts', 'app.module.ts'), 'assets', and 'environments'. The right side of the screen shows the code editor with 'app.component.ts' open. The code defines an AppComponent component with a selector of 'app-root' and templateUrl 'app.component.html'.

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'angularapp';
}
```

Fonte: adaptado do editor de Código *Visual Studio Code v1.56*.

## Referências bibliográficas

W3Schools. **JavaScript and HTML DOM Reference**. 2020.

SILVA, M. S. **JavaScript - Guia do Programador**: guia completo das funcionalidades de linguagem JavaScript. São Paulo: Novatec, 2020. 608p.

## PARA SABER MAIS

Você sabe quais são as diferenças de atributos de ventos do HTML, relacionados ao JavaScript em relação as declarações de modelos do Angular?

Uma declaração de modelo, ou template *statements*, são métodos ou propriedades que podem ser utilizadas no HTML para responder a eventos realizados pelos usuários. Com as declarações de modelos, é possível criar interfaces mais atraentes aos usuários, uma vez que suas ações podem resultar na apresentação de conteúdos

dinâmicos ou em respostas às submissões de formulários, entre outras ações (ANGULAR, 2021).

No exemplo a seguir, a declaração no modelo do método “efetuarCalculo()” aparece entre aspas duplas, sendo atribuído a um valor entre parênteses:

```
<button (click)="efetuarCalculo()">Calcular</button>
```

O trecho “(click) = “efetuarCalculo()”” é representado genericamente por (evento) = “declaração”. No exemplo, quando um usuário clica no botão *Calcular*, o método “efetuarCalculo()”, que está declarado no arquivo do componente em TypeScript (.ts), é chamado.

Você pode usar declaração de modelos com elementos, componentes ou diretivas em resposta a outros eventos. Considerando que são usados os parênteses para indicar um evento no elemento em HTML, no botão, como pode ser visto no trecho anterior, temos os dados sendo enviados de modo unidirecional. Contudo, se diferente das expressões de templates comumente utilizadas em JavaScript.

As declarações de modelos suportam o assinalamento básico, por meio do sinal de igual (“=”), quanto por expressões encadeadas com ponto e vírgula. Adicionalmente, as declarações de modelos no Angular não suportam as expressões permitidas no JavaScript, sendo:

- Palavra reservada para instanciar objetos: *new*.
- Operadores de incremento ou decremento (“++” e “--”, respectivamente).
- Operadores de atribuição, tais como “+=” e “-=”.

- Os operadores de bit a bit (*bitwise*) como “|” e “&”.
- Operador *pipe* “|”.

É importante destacar, ainda, que as declarações possuem contexto, que corresponde a uma parte particular no qual a declaração pertence.

Uma declaração pode se referir apenas a uma declaração em seu contexto, o que ocorre tipicamente na instância do componente. Por exemplo, no código anterior, “efetuarCalculo()” é um método do próprio componente.

A declaração de contexto pode se referir também a uma propriedade que possui seu próprio contexto de template. No exemplo a seguir, mesmo o componente manipulando o evento do método, “salvar()” possui seu objeto de próprio template “\$event” como um argumento. Nas linhas seguintes, o método “excluirHeroi()” obtém a variável do *input*, herói, e “onSubmit()” obtém a referência do template da variável “#formularioHeroi”.

```
<button (click)="salvar($event)">Salvar</button>

<button *ngFor="let heroi of herois" (click)=""
excluirHeroi(heroi)">{{heroi.nome}}</button>

<form #formularioHeroi
(ngSubmit)="onSubmit(formularioHeroi)"> ... </form>
```

Neste exemplo, o contexto do objeto *%event*, *heroi*, e *#formularioHeroi* é o template.

Os nomes de contexto dos *templates* possuem precedência sobre nomes de contexto dos componentes. No código anterior,

“`excluirHeroi(heroi)`”, “`heroi`” é a variável de `input` do template, e não a propriedade do componente “`heroi`”.

Como prática de estudo, indica-se a utilização destas declarações de modelos em seus projetos Angular.

## Referências bibliográficas

ANGULAR. Documentação Oficial Angular.

## TEORIA EM PRÁTICA

Considerando que uma aplicação em Angular utiliza componentes para ser criada e a reutilização de tais componentes é o um dos principais motivadores do uso de tal *framework*, crie um componente de menu que leve ao componente *Calculadora*, especificado na Leitura Digital, e a uma página de boas-vindas. O objetivo é praticar a criação de rotas e os elementos estudados até o momento.

**Para conhecer a resolução comentada proposta pelo professor, acesse a videoaula deste *Teoria em Prática* no ambiente de aprendizagem.**

## LEITURA FUNDAMENTAL

### Indicações de leitura

**Prezado aluno, as indicações a seguir podem estar disponíveis em algum dos parceiros da nossa Biblioteca Virtual (faça o log in por meio do seu AVA), e outras podem estar disponíveis em sites acadêmicos (como o SciELO), repositórios de instituições**

**públicas, órgãos públicos, anais de eventos científicos ou periódicos científicos, todos acessíveis pela internet.**

**Isso não significa que o protagonismo da sua jornada de autodesenvolvimento deva mudar de foco. Reconhecemos que você é a autoridade máxima da sua própria vida e deve, portanto, assumir uma postura autônoma nos estudos e na construção da sua carreira profissional.**

**Por isso, nós o convidamos a explorar todas as possibilidades da nossa Biblioteca Virtual e além! Sucesso!**

## **Indicação 1**

Este livro é dividido em três partes, apresentando uma introdução à JavaScript e JQuery, seguindo para funcionalidades HTML5, Bootstrap e frameworks AngularJS e Angular, possibilitando ao leitor visualizar as diferentes e se aprofundar em diversos conhecimentos sobre o desenvolvimento de aplicações web.

Para realizar a leitura, acesse a plataforma Biblioteca Virtual, acesse o parceiro Minha Biblioteca e busque pelo título da obra no campo de busca.

MATOS, E.; ZABOT, D. **Aplicativos com Bootstrap e Angular:** como desenvolver apps responsivos. São Paulo: Saraiva Educação SA, 2020.

## **Indicação 2**

Neste livro, os autores ensinam HTML, CSS, JavaScript e PHP, porém, mais importante que tais conteúdos, é a possibilidade de entender

como integrar e construir bancos de dados com MySQL, nas aplicações web.

Para realizar a leitura, acesse a plataforma Biblioteca Virtual e busque no parceiro Senac, pelo título da obra campo de busca.

KLACERDA, I. M. F.; OLIVEIRA, A. L. S. **Programando web**: um guia para programação e manipulação de banco de dados. Rio de Janeiro: Senac Nacional, 2013.

## QUIZ

**Prezado aluno, as questões do Quiz têm como propósito a verificação de leitura dos itens *Direto ao Ponto, Para Saber Mais, Teoria em Prática e Leitura Fundamental*, presentes neste Aprendizagem em Foco.**

**Para as avaliações virtuais e presenciais, as questões serão elaboradas a partir de todos os itens do Aprendizagem em Foco e dos slides usados para a gravação das videoaulas, além de questões de interpretação com embasamento no cabeçalho da questão.**

1. Uma aplicação Angular é gerada pela integração e diversos componentes. Sobre as partes que compõem um componente Angular e suas especificidades, assinale a alternativa correta:
  - a. Um template é constituído dos elementos HTML que serão renderizados, sendo mantidos em um arquivo XHTML.
  - b. Um template é constituído de regras e propriedades que estilizam os componentes, sendo mantidos em arquivo CSS.

- c. Um template é constituído de elementos HTML que serão renderizados, sendo mantidos em um arquivo HTML.
  - d. Um arquivo de estilização corresponde ao arquivo que aplicará regras e propriedades do CSS aos elementos mantidos no arquivo TypeScript.
  - e. Um arquivo TypeScript contém o *decorator* @Campos, responsável por registrar e identificar todos os arquivos que definem um componente.
2. Para a integração de um componente Angular em uma aplicação, é necessária a definição de um *selector*. Sobre a definição de tal *selector* de um componente Angular, assinale a alternativa correta:
- a. O *selector* cria tags HTML que serão utilizadas na classe e em seus métodos, no arquivo TypeScript.
  - b. O *selector* recebe informações do *template* HTML que será utilizado pelo componente.
  - c. O *selector* recebe informações de estilização CSS que será utilizado pelo componente.
  - d. O *selector* recebe um nome que permitirá que este seja inserido e referenciado em um componente por meio de uma tag similar às tags HTML.
  - e. O *selector* é opcional, visto que um componente pode ser integrado à aplicação final por meio do arquivo HTML.



## GABARITO

### Questão 1 - Resposta C

**Resolução:** Um template é constituído de elementos HTML que serão renderizados, sendo mantidos em um arquivo

HTML. Já um arquivo de estilização corresponde ao arquivo que aplicará regras e propriedades do CSS aos elementos do template (HTML). Finalmente, um arquivo TypeScript é onde o componente é definido por meio de um *decorator* @ *Component*, registrando a localização dos arquivos (HTML, CSS), além de identificar o selector a ser utilizado para referenciar tal componente na aplicação.

### **Questão 2 - Resposta D**

**Resolução:** O *selector* recebe um nome que permitirá que este seja inserido e referenciado em um componente por meio de uma *tag* similar às tags HTML. Está presente no corpo do *decorator* @*Component*, juntamente com a especificação do *template* e do arquivo CSS, que estilizará o componente.



**TEMA 4**

# **React: desenvolvimento de componentes com o Framework do Facebook**

---

Autoria: Anderson da Silva Marcolino

Leitura crítica: Paulo Henrique Santini



## DIRETO AO PONTO

A utilização de um *framework* facilita o trabalho da equipe de desenvolvedores, além de permitir maior reuso. O principal destaque, ao se utilizar o React, é a sua capacidade de desenvolver aplicações web por meio dos componentes. Devido a sua importância, é necessário saber como é o comportamento de um componente e seu ciclo de vida.

Um componente é integrado a uma página por meio de um seletor. É possível transformar funções em componentes que possam, por meio de tais seletores, serem renderizados nas páginas.

Os passos para se converter uma função em uma classe, que gerará um componente, são:

1. Criar uma classe em JavaScript (EcmaScript 6 ou versões superiores), com o mesmo nome da função e estendendo `React.Component` (é necessário importar o React para o respectivo arquivo).
2. Adicionar um único método chamado `render()`.
3. Colocar o conteúdo da função no método `render()`.
4. Substituir as propriedades que são enviadas como parâmetros para a função, intituladas `props` por `this.props` no corpo do método `render()`.
5. Excluir todo o código sobressalente à declaração da antiga função.

Temos como resultado, algo similar ao código a seguir:

```
class Relogio extends React.Component {  
  render() {
```

```
return (  
  <div>  
    <h1>Olá, Mundo!</h1>  
    <h2>A hora é {this.props.date.  
toLocaleTimeString()}.</h2>  
  </div>  
);  
}  
}
```

No trecho anterior, temos a declaração de uma classe chamada *Relogio*, que herda as propriedades do componente React (`class Relogio extends React.Component`) e, assim, pode utilizar o método de renderização (`render()`), que apresenta um retorno, que é todo o componente criado.

O método `render()` será chamado toda vez que ocorrer uma atualização no valor da propriedade `ali` passada. A renderização ocorrerá para o elemento `<Relogio>` do DOM. Isso garante que os recursos adicionais, como local e os métodos do ciclo de vida, possam ser utilizados. Assim, precisamos passar as propriedades para o componente, o que requer a mudança do código como o apresentado a seguir, considerando o trecho anterior:

```
class Relogio extends React.Component {  
  constructor(props) {  
    super(props);  
  }  
}
```

```
    this.state = {date: new Date()};

}

render() {
  return (
    <div>
      <h1>Olá, Mundo!</h1>
      <h2>A hora é {this.props.date.toLocaleTimeString()}.</h2>
    </div>
  );
}

ReactDOM.render(
  <Relogio />,
  document.getElementById('root')
);
```

As seguintes mudanças foram realizadas:

1. Substituição do `this.props.date` por `this.state.date` no método `render()`;
2. Incluir um construtor que atribua o valor da nova data ao `this.state.date` (`this.state = {date: new Date()}`);
3. Remover a `props` do elemento `<Relogio />`

A Figura 1 apresenta a integração de métodos de ciclo de vida do React ao código anterior, de forma a apresentar o código do componente Relógio com métodos do ciclo de vida do React.

Considerando que nosso componente Relógio deve atualizar a hora, por meio de um temporizador, ao ser renderizado pelo DOM, precisamos, então, montar o componente. Já quando queremos destruir o componente, após seu uso, precisamos desmontá-lo. Para isso, temos dois métodos que fazem parte do ciclo de vida React: `componentDidMount()` e `componentWillUnmount()`.

O método `componentDidMount()` é executado depois que o componente é renderizado no DOM e o método `componentWillUnmount()` é executado assim que o temporizador do relógio é interrompido.

Deste modo, a Figura 1 representa ordenadamente:

1. O `<Relogio/>` é renderizado ao passar pela chamada `ReactDOM.render()` tendo o `this.state` inicializado com um objeto que insere a data atual.
2. React chama `render()` e sabe que deve exibir o componente em tela, atualizando o DOM.
3. Logo que o relógio é inserida no DOM, o React chama `componentDidMount()` e envia a demanda para que o temporizador, por meio do método `tick()` seja chamado a cada segundo.

4. A cada chamada do método `tick()` temos a chamada do método `setState()` com um objeto atualizado com a hora atual, indicando que é necessário ao método `render()` uma atualização e renderização com o novo valor.
5. Ao final, quando o componente deixar de ser exibido, o método `componentWillUnmount()` é chamado para finalizar a chamada do temporizador, o interrompendo.

### **Figura 1 – Componente Relógio**

```

1  class Clock extends React.Component {
2    constructor(props) {
3      super(props);
4      this.state = {date: new Date()};
5    }
6    componentDidMount() {
7      this.timerID = setInterval(
8        () => this.tick(), 1000);
9    }
10   componentWillUnmount() {
11     clearInterval(this.timerID);
12   }
13   tick() {
14     this.setState({date: new Date()});
15   }
16   render() {
17     return (
18       <div>
19         <h1>Hello, world!</h1>
20         <h2>It is {this.state.date.toLocaleTimeString()}.</h2>
21       </div>
22     );
23   }
24 }
25 ReactDOM.render(<Clock />,document.getElementById('root'));
26
27

```

Fonte: adaptado e traduzido de REACT (2021) e capturado do editor de código *Visual Studio Code* versão 1.56.

## **Referências bibliográficas**

REACT. **Documentation**. 2021.



## **PARA SABER MAIS**

Você sabia que é possível utilizar renderização condicional no React?

Com React é possível renderizar componentes distintos após a verificação condicional, similar à condição “se” do JavaScript. Esta condição pode seguir o estado da aplicação.

```
function SaudacaoAoCliente (props) {  
  return <h1>Olá Cliente!</h1>;  
}  
  
function SaudacaoAoConvidado (props) {  
  return <h1>Olá Convidado!</h1>;  
}  
  
function Saudacoes(props) {  
  const estaLogado = props.estaLogado;  
  
  if (estaLogado) {  
    return <SaudacaoAoCliente />;  
  }  
  
  return <SaudacaoAoConvidado />;  
}  
  
ReactDOM.render(  
  <Saudacoes />,  
  document.getElementById('root')
```

```
< Saudacoes estaLogado ={false} />,  
document.getElementById('root')  
);
```

No exemplo, se o usuário está logado, temos uma mensagem específica, indicando a mensagem *Olá Cliente!*, já se o cliente não está logado ainda, então uma mensagem de *Olá Convidado* é exibida. Para fazer o teste no código e verificar o comportamento, é necessário alterar o valor enviado no método `ReactDOM.render()` de `false` para `true`. Com a propriedade sendo passada para o método `Saudacoes`, a condicional é, então, executada e apresentará valores específicos, conforme o valor lógico passado.

Interessante, não? É importante mencionar que estes são só apenas alguns dos recursos existentes o React. Logo, estes e os outros recursos são importantes e merecem ser estudados, principalmente, os fornecidos na documentação oficial, que está sempre atualizada (REACT, 2021).

## Referências bibliográficas

REACT. **Documentation**. 2021.

## TEORIA EM PRÁTICA

Considerando que o React possui a troca de propriedade entre componentes e esta troca é o que permite a composição de componentes mais robustos que, ao final, formam aplicações complexas, crie um componente que atualize, além de inserir uma

tarefa em uma lista, permita a exclusão da tarefa, quando esta for considerada concluída.

**Para conhecer a resolução comentada proposta pelo professor, acesse a videoaula deste *Teoria em Prática* no ambiente de aprendizagem.**

## LEITURA FUNDAMENTAL

### Indicações de leitura

**Prezado aluno, as indicações a seguir podem estar disponíveis em algum dos parceiros da nossa Biblioteca Virtual (faça o log in por meio do seu AVA), e outras podem estar disponíveis em sites acadêmicos (como o SciELO), repositórios de instituições públicas, órgãos públicos, anais de eventos científicos ou periódicos científicos, todos acessíveis pela internet.**

**Isso não significa que o protagonismo da sua jornada de autodesenvolvimento deva mudar de foco. Reconhecemos que você é a autoridade máxima da sua própria vida e deve, portanto, assumir uma postura autônoma nos estudos e na construção da sua carreira profissional.**

**Por isso, nós o convidamos a explorar todas as possibilidades da nossa Biblioteca Virtual e além! Sucesso!**

### Indicação 1

A obra tem início com uma introdução concisa sobre os padrões XHTML, CSS e JavaScript. Em seguida, traz lições sobre o desenvolvimento avançado do lado do cliente, apresentando padrões como DOM, XML, Ajax, JSON e outras tecnologias Rich

Internet Applications. Por fim, são abordados os aspectos do lado do servidor, incluindo serviços Web, bancos de dados, PHP, Ruby on Rails, ASP .NET, JavaServer Faces e serviços Web. Isso expandindo o conhecimento de outras tecnologias e aprimorando o processo de desenvolvimento web.

Para realizar a leitura, acesse a plataforma Biblioteca Virtual, acesse o parceiro Pearson/ Biblioteca Virtual 3.0 e busque pelo título da obra campo de busca.

**DEITEL, P. J.; DEITEL, H. M. *Ajax, rich internet applications e desenvolvimento Web para programadores*.** São Paulo: Pearson Prentice Hall, 2009.

## **Indicação 2**

Neste livro, diversos conceitos web são tratados, mas o foco são as interfaces e como podem ser criadas. No contexto de React, que trata de um *framework front end*, saber como estrutura e criar tais interfaces, é essencial e este livro trará tudo que é necessário para criar interfaces de sucesso.

Para realizar a leitura, acesse a plataforma Biblioteca Virtual, acesse o parceiro Pearson/ Biblioteca Virtual 3.0 e busque pelo título da obra campo de busca.

**SEGURADO, V. S. *Projeto de interface com o usuário*.** São Paulo: Pearson Prentice Hall, 2016.



## QUIZ

**Prezado aluno, as questões do Quiz têm como propósito a verificação de leitura dos itens *Direto ao Ponto, Para Saber Mais, Teoria em Prática e Leitura Fundamental*, presentes neste Aprendizagem em Foco.**

**Para as avaliações virtuais e presenciais, as questões serão elaboradas a partir de todos os itens do Aprendizagem em Foco e dos slides usados para a gravação das videoaulas, além de questões de interpretação com embasamento no cabeçalho da questão.**

1. O *framework* React se baseia na construção de componentes para a criação de aplicações web. Sobre a criação e conversão de uma função em JavaScript para componentes React, assinale a alternativa correta:
  - a. Um componente React é integrado e renderizado, em uma página, por meio de um conversor.
  - b. Na conversão de uma função para um componente, o conteúdo da função é inserido no corpo do método render().
  - c. Em um componente React, a função será traduzida em um método responsável por renderizar o conteúdo, sendo este o `document.getElementById('root')`.
  - d. O método render() não precisar de um retorno para que o componente seja apresentado ao usuário.
  - e. A função JavaScript deve ser transcrita em um componente, deverá ser escrita, gerando uma classe que deve estender React.render().

2. As aplicações React e seus componentes apresentam métodos específicos que são responsáveis pelo ciclo de vida dos componentes e seus comportamentos. Considerando tais métodos, assinale a alternativa correta:
- a. O método `componentWillUnmount()` é executado no início do ciclo do componente React.
  - b. Em `componentWillUnmount()`, deve-se indicar métodos ou executar ações que preparem a execução dos elementos e valores de estado para que a aplicação possa ser renderizada.
  - c. O método `componentDidMount()` garante que os valores iniciais da aplicação e das propriedades dos componentes estejam configurados corretamente, necessitando chamar especificamente o método `render()` como apoio.
  - d. O método `render()` é o método responsável por renderizar, de fato, o componente, contudo pode ser suprimido, sempre que conveniente..
  - e. O método `componentDidMount()` é executado depois que o componente é renderizado no DOM.



## GABARITO

### Questão 1 - Resposta B

**Resolução:** Um componente React é integrado e renderizado em uma página por meio de um seletor. Quando uma função é traduzida em um método responsável por renderizar o conteúdo, tal método é o `render()`, que precisa, obrigatoriamente, de um retorno para que o componente seja apresentado ao usuário. Finalmente, para que uma função

JavaScript seja transcrita em um componente, deverá ser transformada em uma classe que estenderá React.Component.

## Questão 2 - Resposta E

**Resolução:** O método `componentWillUnmount()` é executado no final do ciclo do componente React, neste método, deve-se indicar métodos ou executar ações que finalizam a execução dos elementos e valores de estado da aplicação. Já o método `componentDidMount()` garante que os valores iniciais da aplicação e das propriedades dos componentes estejam configurados corretamente, não dependendo do método `render()` como apoio, já que este método é o responsável por inicializar o componente e sinalizar para o método `componentDidMount()` para inicializar os componentes e seus respectivos estados e valores, para que o comportamento esperado do mesmo seja executado.



---

**BONS ESTUDOS!**