Desenvolvimento Reativo

Capítulo 2. *Single Page Applications* (SPA's)

Prof. Raphael Gomide

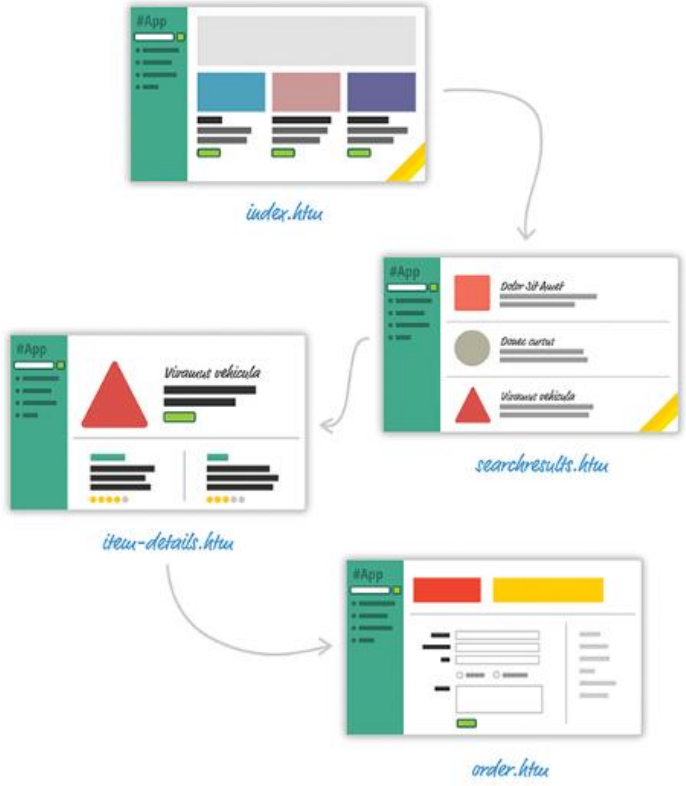
Aula 2.1. Introdução às SPAs

Nesta aula

- A *web* antes das SPAs.
- *Single Page Applications.*
- Construção de SPAs atualmente.

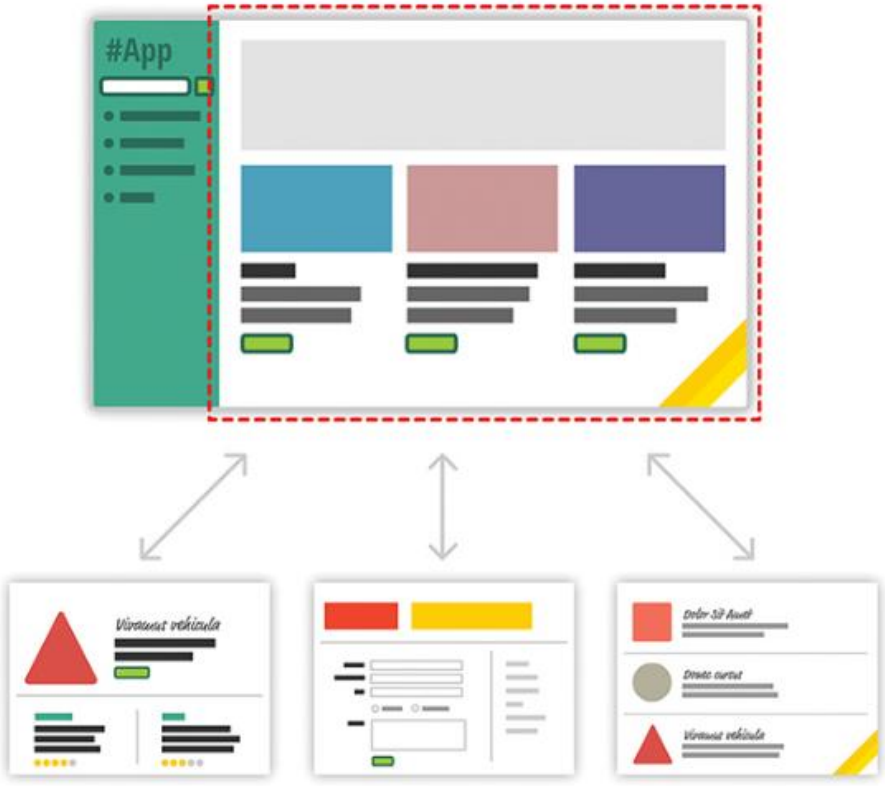
A web antes das SPAs

- Sites com múltiplas páginas (MPA – *Multiple Page Application*).
- Cada página tem o seu escopo e estado.
- Cada ação necessita de uma nova requisição ao navegador.
- *Refresh* a cada troca de página.
- Lentidão durante a navegação.
- Prejudica a experiência do usuário.
- Apesar dos problemas, perdura e é suficiente em alguns casos.
- Demonstração: G1.

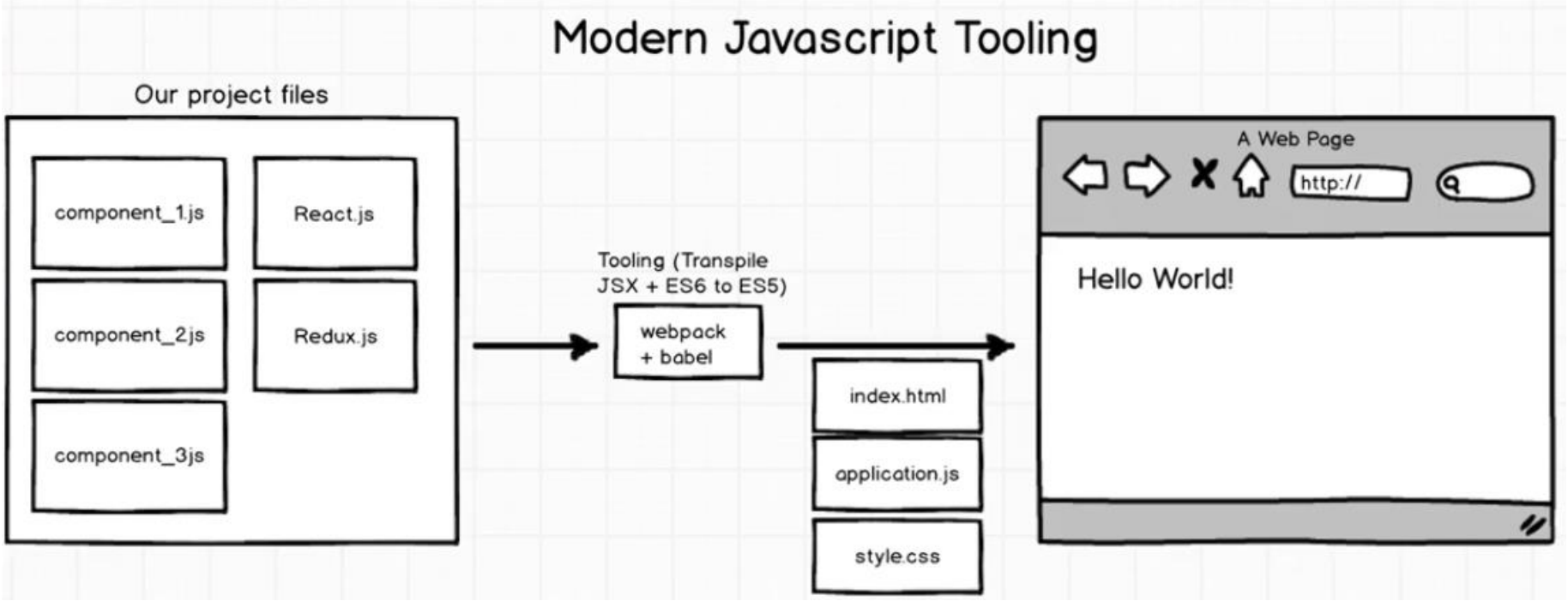


Single Page Applications

- Uma única página é carregada.
- As demais transições e requisições ao servidor são feitas sem o *refresh* do navegador.
- Semelhantes às aplicações *desktop*.
- Melhora a experiência do usuário, tornando-a mais fluida.
- A lentidão é mais comum somente na inicialização.
- Demonstração: Google Inbox.



Construção de SPAs atualmente.



- Melhoramos a experiência do **desenvolvedor** e...
- ... não necessariamente a experiência do **usuário**.

Conclusão

- ☑ MPAs (*Multiple Page Applications*):
  - Sites com múltiplas páginas.
  - *Refresh* no navegador a cada requisição.
  - Tendem a ser mais lentas na utilização.
  - Podem piorar a experiência do usuário.
- ☑ SPAs (*Single Page Applications*):
  - Sites com somente uma página.
  - Após a carga inicial, não há mais o *refresh n*o navegador.
  - Tendem a ser lentas somente na inicialização.
  - Melhoram a experiência do desenvolvedor.
  - Podem melhorar a experiência do usuário.

Próxima aula

- Conceitos importantes sobre *Single Page Applications*.

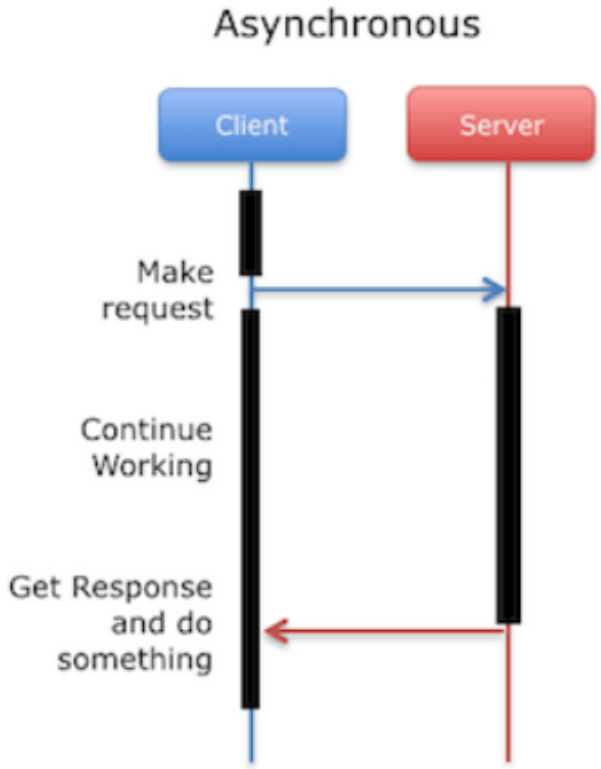
Aula 2.2. Conceitos importantes sobre SPAs

Nesta aula

- Requisições assíncronas.
- Manipulação do DOM (*Document Object Model*).
- Roteamento.

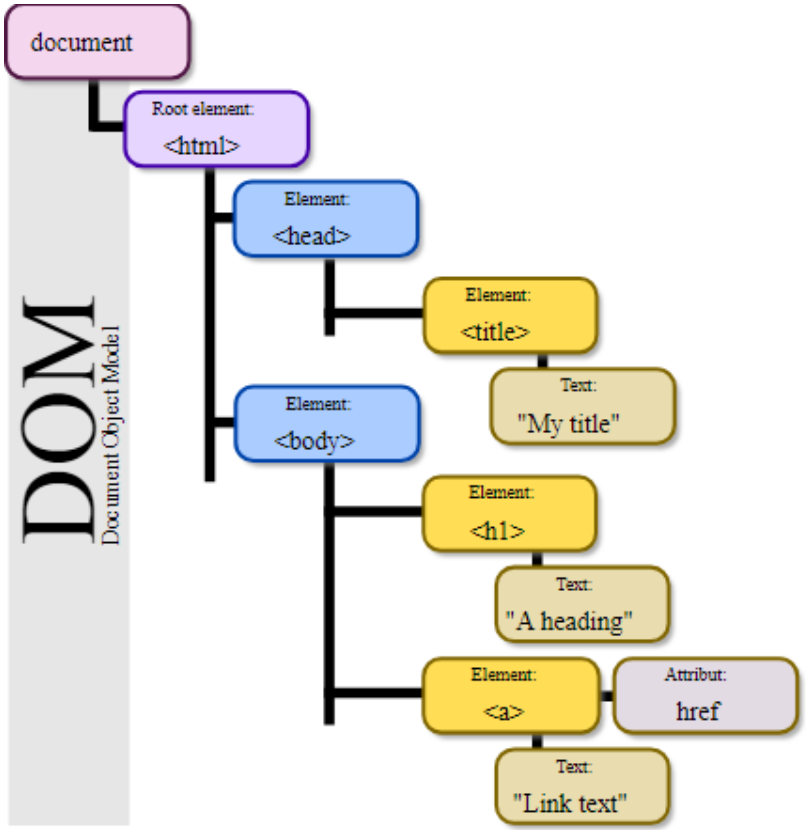
Requisições assíncronas

- Essenciais para o funcionamento de SPAs.
- Não bloqueiam o funcionamento do *site.*
- Manipulação do DOM sem *refresh* do navegador.
- Com JavaScript, é possível através de:
  - *XMLHttpRequest;*
  - *Callbacks;*
  - *Promise;*
  - *Async/await;*
  - *Observables* (RxJS);



Manipulação do DOM (*Document Object Model*)

- SPAs manipulam o DOM constantemente.
- A inserção/atualização/remoção de elementos é considerada cara (lenta).
- São necessárias estratégias/algoritmos para manipular eficientemente o DOM.
- Um DOM bem manipulado melhora a experiência do usuário consideravelmente.



Roteamento

- SPAs devem simular a navegação entre *links* sem gerar o *refresh* nas páginas.
- Isso é feito com o processo de **roteamento**.
- Resumidamente:
  - Os links que apontam para outras “páginas” do site são recarregados instantaneamente, sem *refresh*.
  - Para isso, essa navegação é capturada pelo sistema de roteamento, que então faz a manipulação do DOM necessária, evitando o *refresh*.
- Exemplo de utilização: o usuário normalmente utiliza o botão “Voltar” – isso deve continuar funcionando.
- Atualmente, SPAs podem utilizar a *History API,* por exemplo.

Conclusão

☑ Conceitos importantes de SPAs:

- Requisições assíncronas: o *site* continua funcionando e dando *feedback* ao usuário durante as requisições;
- Manipulação do DOM: o DOM deve ser manipulando eficientemente, pois é uma operação cara (lenta);
- Roteamento: a SPA deve simular a troca de links sem ocasionar a ação de *refresh* no navegador.

Próxima aula

- Introdução aos frameworks de SPAs.

Aula 2.3. Introdução aos *frameworks* de SPAs.

Nesta aula

- Motivação.
- Plataforma de desenvolvimento – Node.js.

Motivação

- Requisições assíncronas.
- Manipulação eficiente do DOM.
- Roteamento.
- Validação de formulários.
- Acesso às APIs do *backend*.
- Componentização.
- Modularização.
- Experiência do desenvolvedor.
- Etc.

Plataforma de desenvolvimento – Node.js

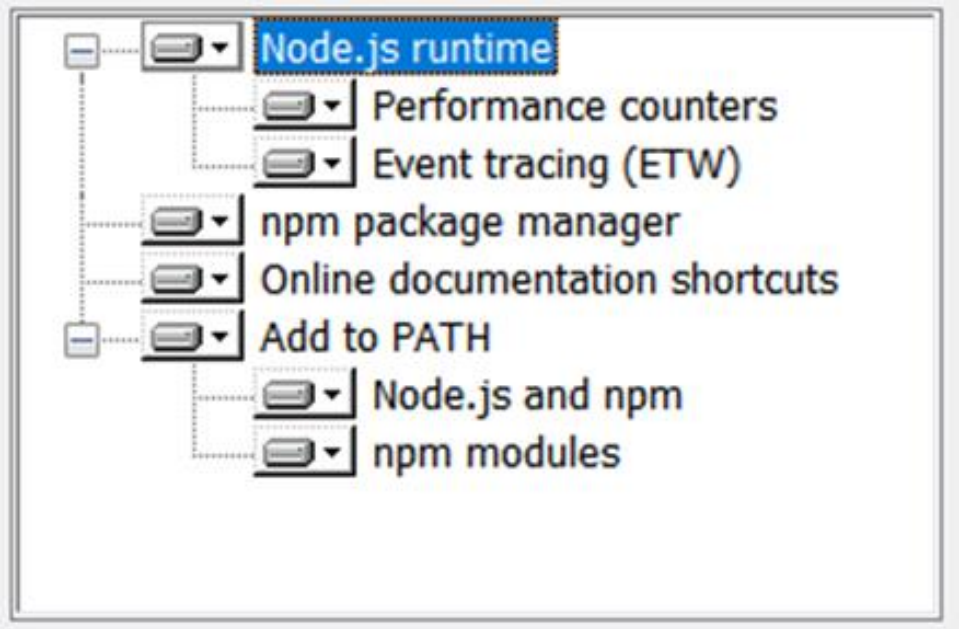
- Ambiente de *runtime* JavaScript:
  - Utiliza a *engine* JavaScript V8 do Google.
  - Multiplataforma.
  - Permite a execução de JavaScript no servidor.
  - É bastante utilizado como ecossistema de desenvolvimento de diversas tecnologias.
  - Possui o maior repositório de bibliotecas *open source* mundial.
  - As bibliotecas são mais conhecidas como *pacotes (packages).*
  - https://nodejs.org.

Instalação – Parte I

- Recomenda-se a instalação da versão LTS (*Long Term Support*):
  - É importante que o comando node seja acessível de qualquer pasta (global).
  - Atualmente:

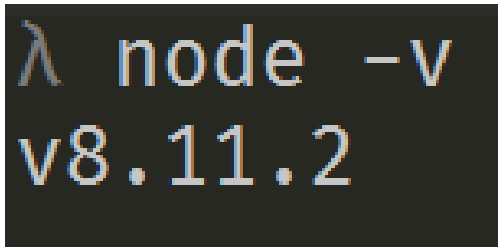
Instalação – Parte II

- [Windows] Durante a instalação, certifique-se de que as opções em destaque estejam **marcadas** antes de prosseguir:



Instalação – Parte III

- Teste após a instalação.
  - Utilize o terminal.
  - Execute o seguinte comando: “node -v”.
  - É importante que o comando seja global, ou seja, funcione independentemente da pasta em que o usuário está situado.



npm (*Node Package Manager*)

- Gerenciador de pacotes do Node.js:
  - Repositório imenso de pacotes *open source.*
  - Localmente, os pacotes são hospedados na pasta *node_modules* de cada projeto.
  - Instalação global x local.
  - Configuração do projeto – arquivo “package.json”.
  - Comando *npm.*

Exemplos de comandos importantes (terminal)

Comando	Descrição
npm i nome_do_pacote -g	Instalação **global** do pacote
npm i nome_do_pacote	Instalação **local** do pacote
npm i nome_do_pacote --save	Instalação **local** do pacote (Dependência de projeto)
npm i nome_do_pacote --save-dev	Instalação **local** do pacote (Dependência de desenvolvimento)
npm r nome_do_pacote (-g --save --save-dev)	Exclusão do pacote
npm view nome_do_pacote	Informações importantes (versões)
npm i –g npm	Atualiza o próprio npm

Conclusão

- ☑ Frameworks para SPAs surgem frequentemente para resolver os mesmos problemas de formas diferentes (ou não).
- ☑ Node.js:
  - Ambiente de execução JavaScript.
  - Ecossistema de desenvolvimento.
  - Utilizado por várias tecnologias, incluindo o Cordova e o Ionic.
- ☑ NPM:
  - Gerenciador de pacotes (bibliotecas) do Node.js.

Próxima aula

- Capítulo 3 – Introdução ao Angular.