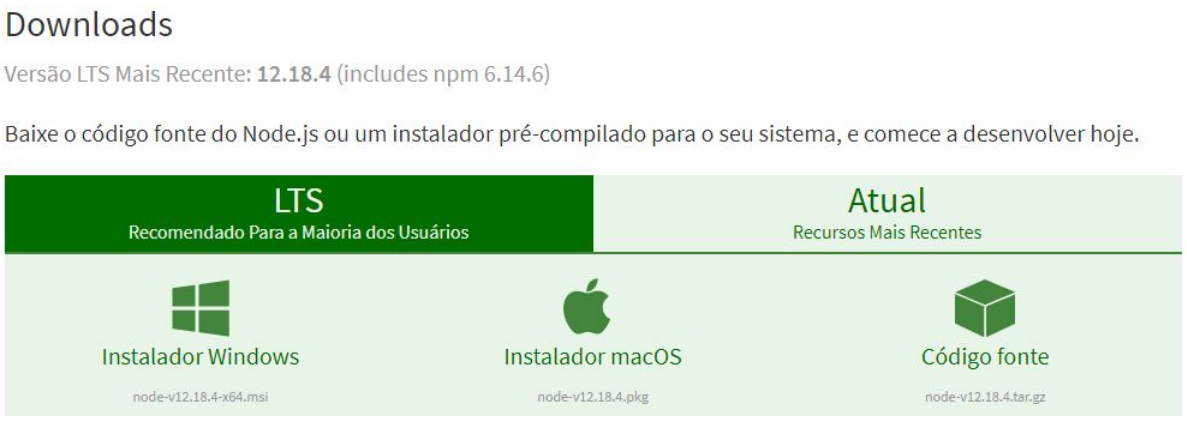
COMO CONFIGURAR O AMBIENTE PARA USAR O JAVASCRIPT ES6+

Instalar dependências

- Instalar o NodeJS

Link para dowload: https://nodejs.org/pt-br/download/



> Para garantir que foi instalado corretamente, abra o prompt de comando (cmd) e digite o comando:

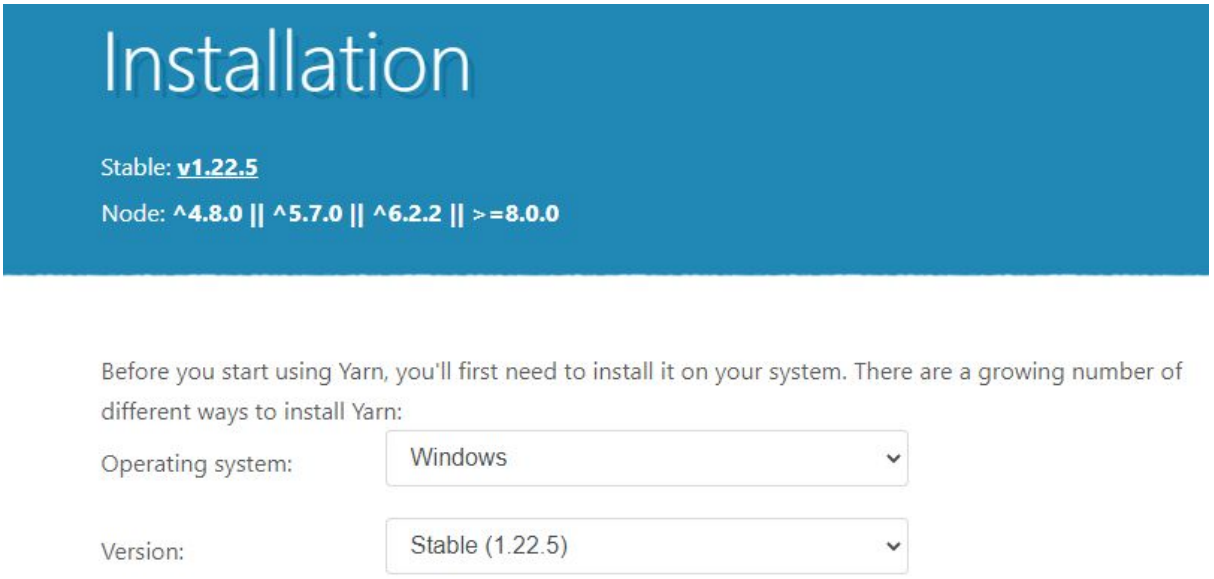
node -v

Esse comando serve para verificar se a dependência está instalada e consultar a versão.

```
C:\Users\rapma>node -v
v12.18.3
```

- Instalar o Yarn

Link para download: https://classic.yarnpkg.com/en/docs/install/#windows-stable (caso tenha dúvidas, leia atentamente ao passo a passo de instalação no próprio site)



O Yarn é um gerenciador de pacotes do JavaScript, ele será utilizado para instalar diversas ferramentas para o projeto e utilizá-las.

> Para garantir que foi instalado corretamente, abra o prompt de comando (cmd) e digite o comando:

yarn -v

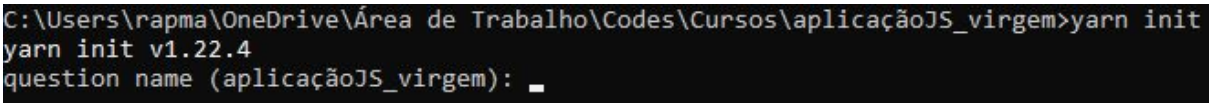
Esse é o mesmo caso que o comando anterior.

Observação: a versão que você instalar não deve necessariamente estar igual às das imagens.

Iniciando a aplicação

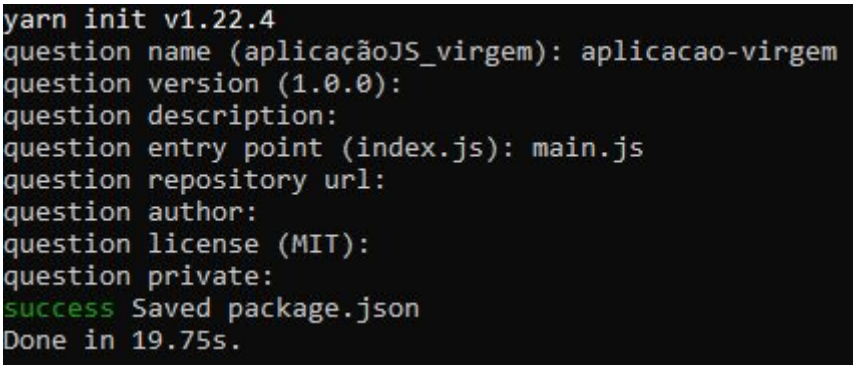
Para dar início ao projeto, é necessário executar alguns comandos para que a aplicação seja criada.
Abra o prompt e vá para a pasta raíz do projeto, a qual você deseja instalar a aplicação (**NÃO SIMPLESMENTE ABRA O PROMPT E COMECE A EXECUTAR OS COMANDOS**).

- O primeiro comando serve para iniciar a aplicação e habilitar novos comandos:
  yarn init



Após executar esse comando, é necessário nomear a aplicação. O nome deve seguir algumas regras:
> Não pode ter espaços (substitua por "-" ou "_");
> Não pode ter caracteres especiais;
> Não pode ter letras maiúsculas.

Após dar nomear o projeto, tecle ENTER para todas as perguntas, mostradas abaixo:



Executado esse comando, um arquivo “package.json” será adicionado à raiz do projeto. Esse arquivo irá armazenar as informações de dependências da aplicação.

- A primeira dependência a ser instalada é o core do Babel, que instalamos com o comando:
  yarn add @babel/core

```
C:\Users\rapma\OneDrive\Área de Trabalho\Codes\Cursos\aplicaçãoJS_virgem>yarn add @babel/core
yarn add v1.22.4
```

- A segunda dependência é o cli do Babel, que instalamos com o comando:
  yarn add @babel/cli

```
yarn add v1.22.4neDrive\Área de Trabalho\Codes\Cursos\aplicaçãoJS_virgem>yarn add @babel/cli
[1/4] Resolving packages...
```

(deu algum bug no começo da linha, apenas ignorem...)

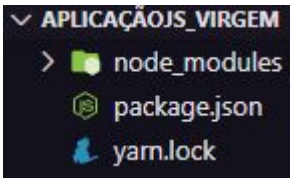
Essa dependência dará a possibilidade de trabalhar com a interface de linha de comando do Babel. Isso nada mais é que ter a possibilidade de executar comandos pelo terminal.

- A terceira dependência é o preset-env do Babel, que instalamos com o comando:
  yarn add @babel/preset-env

```
yarn add v1.22.4neDrive\Área de Trabalho\Codes\Cursos\aplicaçãoJS_virgem>yarn add @babel/preset-env
[1/4] Resolving packages...
[2/4] Fetching packages...
```

O Babel possuí vários presets, esse em específico serve para identificar o ambiente em que estamos trabalhando, e com isso, converter as features do ES6+ para uma forma que o seja melhor interpretada.

Após executar os comandos, os seguintes arquivos serão adicionados à pasta raiz do projeto:



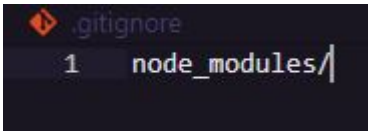
O arquivo yarn.lock é uma forma de cache do yarn, é aconselhado que nunca altere nenhum dado desse arquivo.

A pasta node_modules armazena todas as dependências instaladas. Também é aconselhado que nunca altere nenhum arquivo dessa pasta.

Para questões de desempenho, e se você armazenar essa aplicação em alguma ferramenta de controle de versão, nesse caso o GitHub, é aconselhado também que você crie um arquivo chamado ".gitignore" e adicione a pasta node_modules, já que os arquivos dessa pasta são pesados e não tem nenhum motivo para enviá-los à nuvem.

> Crie um arquivo dentro da pasta raiz chamado **.gitignore**

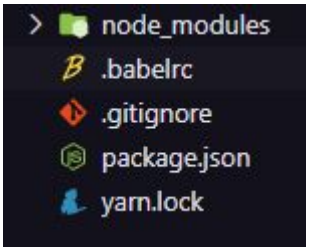
> Em seguida, adicione a pasta node_modules no arquivo



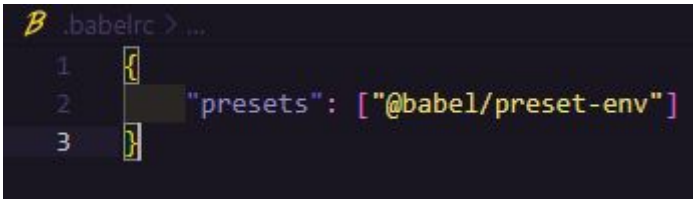
Lembrando: só faça isso se você for utilizar alguma ferramenta de controle de versão, caso contrário, não precisa fazer isso.

Configurando o Babel

Na raiz do projeto, crie um arquivo chamado **.babelrc**. O Babel será responsável por transformar as features do ES6+ em uma forma que o ambiente trabalhado possa interpretar.



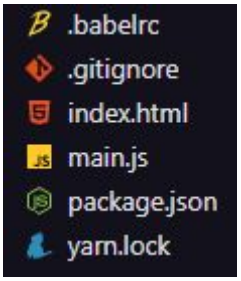
Criado o arquivo, vamos configurá-lo:



São apenas esses os códigos necessários no arquivo.

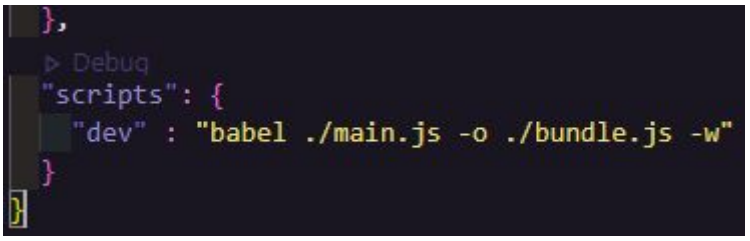
Calma, já estamos acabando....

Concluído os passos anteriores, crie um arquivo **.html** e um **.js**



Agora o último passo é configurar um script para executar o Babel para fazer a conversão dos códigos.

> Vá no arquivo **package.json** e adicione uma nova propriedade com algumas configurações, como feito abaixo:



(não esqueça de colocar a vírgula no fim da propriedade que já está acima para não dar erro...)

Esse script **dev** será utilizado no terminal para executar o Babel, todos os seus códigos (no **main.js** ou no arquivo .js que você selecionar) serão reescritos de uma forma melhor interpretada para o ambiente que você utilizará. Todos os códigos irão para o arquivo **bundle.js**.
O **-w** serve como um observador, é uma tag aplicada no comando para que o Babel fique "observando" as atualizações que você faz no arquivo e atualize o bundle.js assim que você der ctrl+s. Isso é extremamente útil pois exclui a necessidade de executar o comando sempre que você fizer uma atualização no código.

AGORA ESTÁ TUDO PRONTO!
Só uma coisa simples antes de finalizar, o arquivo bundle.js precisa ser importado no html (ou qualquer outra linguagem que você utilizar).

```html
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Aplicação Virgem</title>
</head>
<body>
    <script src="./bundle.js"></script>
</body>
</html>
```

Aqui é apenas um exemplo, mas você pode aplicar a tag script onde bem entender.

P.S.: Caso você tenha esquecido como criar um arquivo .html, assim que abrir o arquivo, digite **html:5** e tecle ENTER para que a estrutura básica do HTML seja criada.

P.S.(2): **Não vai dar uma de louco e codar no bundle.js!** Os códigos têm de ser escritos no main.js, ou seja lá qual o arquivo que você selecionou. O bundle.js nada mais é que a versão melhor interpretável do código para o ambiente. Depois de codar um pouco, olha a loucura que é os códigos no bundle.js.

P.S.(3): Esqueci de falar como faz para executar o comando. É bem simples, na real. Abra o prompt e vá até a pasta raiz da aplicação e execute o comando **yarn dev**. Enquanto você estiver codando, deixe o terminal aberto para o script continuar observando e atualizando o bundle.js.

P.S.(4): A melhor parte disso tudo é que dá pra codar no Visual Studio Code ♥. Não existe nada melhor que isso.

PS5: **VAI VIR POR R$5000 ESSA POR**!! PU** QUE PAROLA**

Considerações finais

Espero mesmo que esse guia tenha te ajudado a configurar o ambiente para utilizar da maravilha que é o ES6+. Umas coisas sensacionais são as features **Template Literals** e as **Arrow Functions**, essas são features incríveis mas que não existiam pré ES6.
Caso tenha dado algum erro ou algo não esteja funcionando, vem de Hiroshima e Nagazap que eu, na medida do possível, tento te ajudar ♥.

Caso não consiga de nenhuma forma criar o projeto por si só, vem de Zap (e eu truco na sua lata marreco) que eu te envio o projeto virgem que criei neste tutorial.

Agradecimentos

Primeiramente quero agradecer a Rocketseat por existir e ter feito um curso gratuito maravilhoso de JavaScript.
Quero agradecer meu gato, meu cachorro, meu papagaio, minha família e meus amigos.
Um big beijo e seja feliz codando em JavaScript.

Na filosofia da suavidade que a gente ganha a vida.

Autor: Diego de Oliveira Freitas