

**CENTRO UNIVERSITÁRIO
INSTITUTO DE EDUCAÇÃO SUPERIOR DE BRASÍLIA – IESB**



Lógica de Programação

MICHEL EMERSON BARROS COSTA, MSC.

1. Configuração do Ambiente

- **Instalação do Node, Instalação do VSCode, Instalação do Code Runner para VSCode).**

2. Estrutura Léxica

- **Caracteres, comentários, literais, identificadores, palavras reservadas, forma de escrita (ponto e vírgula).**

➤ Instalações

▪ Node.js:

- ✓ É uma plataforma construída sobre o motor JavaScript do Google Chrome para facilmente construir aplicações de rede rápidas e escaláveis;
- ✓ Pode ser definido como um ambiente de execução Javascript server-side. É possível criar aplicações Javascript para rodar como uma aplicação stand-alone em uma máquina, não dependendo de um browser para a execução, como estamos acostumados;
- ✓ Node.js é um interpretador de JavaScript assíncrono com código aberto orientado a eventos, criado por Ryan Dahl em 2009, focado em migrar a programação do Javascript do cliente (frontend) para os servidores, criando aplicações de alta escalabilidade (como um servidor web), manipulando milhares de conexões/eventos simultâneas em tempo real numa única máquina física.
- ✓ O Node.js (ambiente de execução Javascript no servidor) foi implementado baseado no interpretador V8 JavaScript Engine, com desenvolvimento mantido pela fundação Node.js em parceria com a Linux Foundation.



➤ Instalações

▪ Node.js:

✓ Instalação:

1. Primeiro passo, acesse o site oficial: <http://nodejs.org> e clique em Download, para usuários do Windows e MacOSX, basta baixar os instaladores dessa página e instalar normalmente. Para quem já utiliza Linux ou algum Package Manager, acesse esse link <https://github.com/joyent/node/wiki/Installing-Node.js-via-package-manager> que é referente as instruções de como instalá-lo em diferentes distribuições Linux que utilizam package managers;
2. Abra o seu terminal console ou prompt de comando para digitar o comando: `node -v`;
3. Para testar digite na aba terminal do Visual Code:
 - `node -v` (deve aparecer a versão do programa) e `npm -v` (deve aparecer a versão do programa)... Obs.:Node --version e Npm -- version (quando instala o node.js o npm que é o gerenciador de pacotes é instalado também).
 - Execute o comando `npm install prompt-sync` (este comando deve ser executado no diretório onde você salva os seus arquivos/programas javascript)... este comando deve ser executado na aba terminal
 - Depois vá no menu opção: File - > Preferences -> settings -> Extensios -> Run Code configuration === dentro desta opção marcar o box Run in Terminal

➤ Instalações

▪ VsCode:

- ✓ Lançado pela Microsoft 2015 é um editor de código destinado ao desenvolvimento de aplicações web chamado Visual Studio Code (VSCode);
- ✓ O VSCode atende a uma quantidade enorme de projetos (ASP .NET, Node.js) e oferece suporte para mais de 30 linguagens de programação, como JavaScript, C#, C++, PHP, Java, HTML, R, CSS, SQL, Markdown, TypeScript, LESS, SASS, JSON, XML e Python, assim como muitos outros formatos de arquivos comuns;
- ✓ Instalação:
 1. Acesse o site oficial do Visual Studio Code <https://code.visualstudio.com/> e faça o download;
 2. Clique em “Executar” e siga os passos indicados;



➤ Instalações

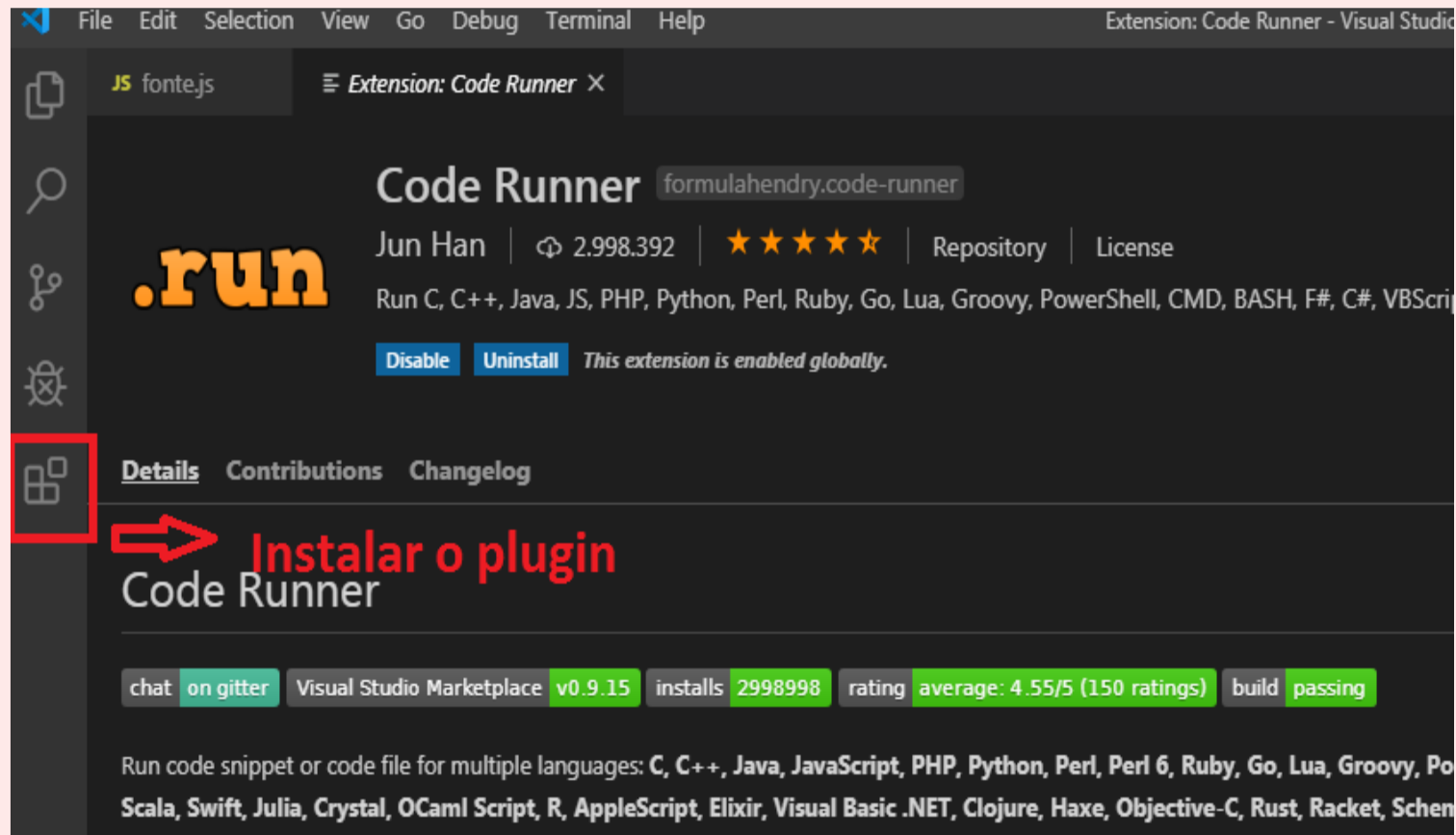


▪ Code Runner:

- ✓ O Code Runner 2 permite que você edite e execute código em qualquer linguagem de programação.
- ✓ O objetivo do app é facilitar a escrita e o teste de códigos, executar scripts, trabalhar com algoritmos ou simplesmente experimentar uma nova linguagem de codificação ou script;
- ✓ foi construído com base no princípio de que você deve ser capaz de executar seu código instantaneamente, em qualquer linguagem.
- ✓ Instalação:
 1. Acesse o site oficial do Code Runner
<https://marketplace.visualstudio.com/items?itemName=formulahendry.code-runner> e faça o download;
 2. Clique em “Install” e siga os passos indicados;
 3. Execute Ctrl+Shift+P -> escreva Code -> Execute -> Shell Command install ‘code’ command in PATH.

➤ Instalações

▪ Code Runner:



sdfsd

➤ Ambiente

- Organização dos arquivos
 1. Crie uma pasta da disciplina em Meus Documento. Ex: LogProg;
 2. Abra o VS Code, vá em File – Open Folder (Selecione o diretório e abra a pasta LogProg;
 3. Organize os arquivos por aula, para tanto clique na opção criar nova pasta dentro do diretório;
 4. Para criar os arquivos basta criar na opção criar arquivo (não esqueça de informar a extensão do arquivo).

➤ Organização

- Uma linha de código representa uma sentença de código, pode ou não terminar com ponto e vírgula (a utilização na sentença é opcional):
 - Ex: `console.log("Sentença de Código")`
- O código de Javascript é organizado em sentença de código e um código também é organizado por bloco de código.
 - Ex: `{ }`
- Javascript é uma linguagem na qual você pode programar procedural, orientada a objetos, funcional;
- O bloco pode estar associado a uma função, classe, controle, repetição... o papel de um bloco é de agrupar as sequências... posso ter um bloco dentro do outro.

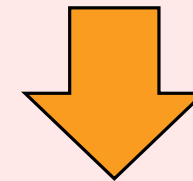
➤ Organização

▪ Indentação

- ✓ Em ciência da computação, indentação é um termo aplicado ao código fonte de um programa para ressaltar ou definir a estrutura do algoritmo.
- ✓ Na maioria das linguagens de programação, a indentação é empregada com o objetivo de ressaltar a estrutura do algoritmo, aumentando assim a legibilidade do código.
- ✓ Indentação é o espaçamento aplicado no início das linhas de código que ajuda a manter uma hierarquia visual. Na maioria das linguagens esse não é um elemento que impede o código de ser compilado, mas manter o código indentado é fundamental para facilitar sua leitura.

SEM Indentação

```
1  <?php
2  if ($nota >= 7) {
3  echo 'Você foi aprovado!';
4  } else { if ($nota > 3) {
5  echo 'Você precisa fazer prova final!';
6  } else {
7  echo 'Você foi reprovado!';
8  }
9  }
```



COM Indentação

```
1  <?php
2  if ($nota >= 7)
3  {
4      echo 'Você foi aprovado!';
5  }
6  else
7  {
8      if ($nota > 3)
9      {
10         echo 'Você precisa fazer prova final!';
11     }
12     else
13     {
14         echo 'Você foi reprovado!';
15     }
16 }
```

➤ Organização

- Exercício (Aula02_Exercicio01)
- JS

```
console.log(  
    "Sentença de código")  
{  
    {  
        console.log("Olá");  
        console.log('Mundo!') // Padrão do curso  
    }  
}
```

- HTML

```
<!DOCTYPE html>  
<html lang="pt-br">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <meta http-equiv="X-UA-Compatible" content="ie=edge">  
    <title>Meu primeiro Programa... </title>  
</head>  
<body>  
    <h1>Olá, mundo!!!</h1>  
</body>  
</html>
```

➤ Executar JavaScript

- Podemos utilizar ferramentas on-line, Vs-Code, Browse, entre outros;
- Alternativas para execução: <https://repl.it/>, <https://jsfiddle.net/>, F12 no console do Browse

➤ Estrutura Léxica

▪ Conjunto de caracteres:

- ✓ Os programas JavaScript são escritos com o conjunto de caracteres Unicode. Unicode é um superconjunto de ASCII e Latin-1 e suporta praticamente todos os idiomas escritos usados hoje;

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

Source: www.LookupTables.com

- ✓ JavaScript é uma linguagem que diferencia letras maiúsculas de minúsculas;
- ✓ JavaScript ignora os espaços que aparecem entre sinais em programas. De modo geral, JavaScript também ignora quebras de linha;
- ✓ você pode formatar e endentar os programas de um modo organizado e harmonioso.

➤ Estrutura Léxica

▪ Comentários:

- ✓ JavaScript aceita dois estilos de comentários. Qualquer texto entre `//` e o final de uma linha é tratado como comentário e é ignorado por JavaScript. Qualquer texto entre os caracteres `/*` e `*/` também é tratado como comentário; esses comentários podem abranger várias linhas, mas não podem ser aninhados.

▪ Literais:

- ✓ Um literal é um valor de dados que aparece diretamente em um programa. Por exemplo:

```
12           // O número doze
1.2          // O número um ponto dois
"hello world" // Uma string de texto
'Hi'         // Outra string
true         // Um valor booleano
false        // O outro valor booleano
/javascript/gi // Uma "expressão regular" literal (para comparação de padrões)
null         // Ausência de um objeto
```

➤ Estrutura Léxica

▪ Identificadores e palavras reservadas

- ✓ Um identificador é simplesmente um nome. Em JavaScript, os identificadores são usados para dar nomes a variáveis e funções e para fornecer rótulos para certos laços no código JavaScript. Um identificador JavaScript deve começar com uma letra, um sublinhado (_) ou um cifrão (\$);
- ✓ Por portabilidade e facilidade de edição, é comum usar apenas letras e dígitos ASCII em identificadores;
- ✓ JavaScript reserva vários identificadores como palavras-chave da própria linguagem. Você não pode usar essas palavras como identificadores em seus programas:

<code>break</code>	<code>delete</code>	<code>function</code>	<code>return</code>	<code>typeof</code>
<code>case</code>	<code>do</code>	<code>if</code>	<code>switch</code>	<code>var</code>
<code>catch</code>	<code>else</code>	<code>in</code>	<code>this</code>	<code>void</code>
<code>continue</code>	<code>false</code>	<code>instanceof</code>	<code>throw</code>	<code>while</code>
<code>debugger</code>	<code>finally</code>	<code>new</code>	<code>true</code>	<code>with</code>
<code>default</code>	<code>for</code>	<code>null</code>	<code>try</code>	

➤ Estrutura Léxica

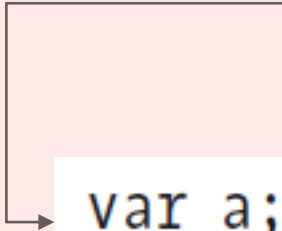
▪ Pontos e vírgulas opcionais

- ✓ JavaScript usa o ponto e vírgula (;) para separar instruções;
- ✓ Você normalmente pode omitir o ponto e vírgula entre duas instruções, caso essas instruções sejam escritas em linhas separadas;
- ✓ JavaScript não trata toda quebra de linha como ponto e vírgula: ela normalmente trata as quebras de linha como pontos e vírgulas somente se não consegue analisar o código sem os pontos e vírgulas.

```
a = 3;  
b = 4;
```

```
a = 3; b = 4;
```

```
var a  
a  
=  
3  
console.log(a)
```



```
var a; a = 3; console.log(a);
```


➤ Estrutura Léxica

- Exercício (Aula02_Exercicio02)

```
// Comentário de uma linha
```

```
console.log('Linha 1')
```

```
/*
```

```
Comentário de  
múltiplas linhas
```

```
*/
```

```
console.log('Linha 2')
```

```
/*
```

```
* Comentário de  
* múltiplas linhas
```

```
*/
```

```
console.log('Linha 3')
```



OBRIGADO!

