Imersão Dev - Aula 2

Rafaella: Olá Devs! **Rafa Ballerini** novamente aqui, muito ansiosa para mostrar para vocês o que teremos no segundo dia de **Imersão Dev**!

Anteriormente, vimos bastante sobre a base da lógica de programação como variáveis e operações, e agora poderemos aprofundar um pouco mais.

Não esqueça que, ao final desta aula, vocês já poderão postar o projeto que fizerem. Iremos adorar ver, seja nas redes sociais ou no GitHub, e isso é uma conquista sua!

Além disso, se tiverem qualquer dúvida, podem postar na nossa comunidade do **Discord**. Inclusive, caso ainda não estejam dentro, convidamos a participar.

Hoje veremos a "mágica" acontecer na página *web*. Então traga o seu **CodePen** e vamos lá!

Guilherme: Boas vindas à segunda aula da Imersão Dev! Sou o **Guilherme Lima** e estou muito empolgado, pois hoje faremos um projeto incrível que podemos de fato usar no dia a dia. Criaremos um **conversor de moedas** em que passaremos um valor em reais, e ele mostrará o valor em dólares, e o contrário também.

Estou também com a Rafa Ballerini e o **Paulo Silveira**!

Rafaella: Olá pessoal!

Paulo: Oi Pessoal! Estou acompanhando vocês no Discord e já compartilhando no primeiro dia, e ficamos muito empolgados! A Rafa deu um *spoiler* na última aula sobre fazer este conversor de moeda não só para treinarmos o que vimos anteriormente, mas daremos um passo além para aprendermos mais sobre as estruturas da programação.

Rafaella: Lembrando o que aprendemos na aula passada, abordamos as **variáveis** e como fazer **operações matemáticas** de soma, subtração e multiplicação.

Nesta aula, uma das coisas mais interessantes que veremos será a **escrita na tela**. Isso também foi um desafio que deixamos na aula passada, que é como fazemos as coisas aparecerem.

Afinal, vimos tudo acontecendo no console, e já sabemos que não o usamos no dia a dia, pois as usuárias e usuários do site não irão usar o console. Então mostraremos como elas inserem os dados diretamente na página e como poderemos imprimi-los também. Vamos lá!

Guilherme: Já estamos com o nosso projeto piloto no CodePen aberto para darmos continuidade. Estou *logado* na minha conta, e vou fazer um *fork* deste projeto.

Paulo: Para lembrar o pessoal, já temos um pouco de código escrito na parte de HTML e CSS, mas o que focaremos nestas primeiras aulas será sempre o JavaScript, que é onde tem mais programação como a maioria das pessoas conhecem.

Até estamos passando o HTML e o CSS, e o conversor de moedas já está vindo com a imagem do **Playstation 5**, que pode ser trocada também. Há diversas *tags* como `<body>`, `background` e `<h1>` que, caso ainda não conheça, não se preocupe, pois nosso foco é na aba JS.

É aí que a Rafa e o Guilherme vão passar o que precisa ser feito, e começaremos pegando o valor do Playstation em dólares para convertê-lo em reais, variando a data da cotação da moeda.

Mostraremos onde é o fork primeiro. Uma vez que já entramos na página, veremos onde clicamos para criar nossa cópia e poder digitar o programa.

Guilherme: Usaremos o projeto piloto para este começo. Como o Paulo falou, já tem um HTML e um CSS já feitos pela equipe de *front-end* da Alura. Pegaremos este projeto e focaremos totalmente na parte da nossa lógica de programação com JavaScript.

Para realizarmos o fork, ou seja, para "garfarmos" este projeto e levá-lo para nossa conta, clicaremos no botão "Fork" ao lado de "Add to Collection" na barra inferior de opções do CodePen.

A partir do momento que fazemos isso, ele fará uma cópia deste projeto para a nossa área de desenvolvimento. Feito isso, poderemos alterar o código e realizar as nossas criações.

Primeiro, se observarmos no HTML, temos várias `<div>` e classes para estilizar esta página e mostrar o conversor de moedas.

Paulo: Agora não tem muito problema quem ainda não conhece bem, é apenas para deixar um pouco mais bonito.

Guilherme: Exatamente, mas tem um ponto que é importante nos atentarmos; a pessoa colocará o valor que quer converter na caixa central da tela, e recebe o nome de *input*. Por exemplo, se digitarmos "10" no campo para fazermos uma conversão de dez dólares para reais, por exemplo, e clicarmos no botão "Converter", de alguma forma precisamos pegar essas informações no lado do HTML e passá-las para o JavaScript.

Na décima primeira linha da aba HTML, encontraremos o `<button>` com uma propriedade chamada `onClick` que já foi passada no código, a qual significa que, quando clicarmos em "Converter", conseguiremos pegar algumas informações do lado do JavaScript e dizer que o botão foi clicado de fato.

Para isso, precisaremos criar um código começando pela palavra function, ou seja, uma **função**. Na programação, este elemento é um compilado de códigos juntos que realizam uma determinada tarefa.

A tarefa que queremos realizar neste caso é converter. Então daremos um nome que será exatamente o mesmo que está no onClick. Os parênteses de Converter() são muito importantes, pois sempre precisamos abri-los e fechá-los, e não podemos usar chaves ou colchetes.

Das várias linhas deste compilado de código que nos permitirá converter de dólares para reais serão escritas dentro serão, identificaremos todo o trecho de código que faz parte desta função abrindo e fechando chaves.

```
function Converter() {}
```

Guilherme: Se pararmos para analisar, é algo que faz sentido. Na função, teremos uma sequência de código que vai realizar uma determinada tarefa, a qual será chamada pelo mesmo nome no onClick, que é o Converter().

Abrimos e fechamos os parênteses () para indicar que se trata de uma função, e em seguida abrimos e fechamos as chaves {}, também conhecidas como "bigodes" ou "bigodinhos".

Dentro das chaves, apertaremos a tecla "Enter" para irmos à linha seguinte. Tudo o que escrevermos dentro será executado quando clicarmos no botão. Para testarmos se isso realmente acontece, deixaremos nosso projeto maior para o visualizarmos e abriremos a aba do console clicando no botão "Console" na barra inferior de opções.

O projeto tem o botão "Converter" e já temos a função Converter(). O CodePen está salvando automaticamente, então encontraremos essa propriedade clicando em "Setings" na barra superior de opções para abrirmos a caixa de "Pen Settings" . Em seguida, clicaremos no item "Behavior" da lista lateral e alteraremos de "On" para "Off" a opção de "Auto-Updating Preview" e deixaremos a "Save Automatically?" como "On" para que o salvamento seja automático mas a atualização da página não.

Paulo: Lembrando que, em todo o projeto, gostaríamos que as alunas e alunos viessem neste mesmo "Settings", acessassem a opção "Pen Details" na lista lateral com os detalhes do esboço, para que então adicionassem "imersaodev" e "alura" ao campo de "Tags". Desta forma, tanto nós quanto outras pessoas conseguirão ver os projetos marcados e interagir, como dar um *like* em um sistema de votação, o que é muito legal.

Guilherme: Isso mesmo! Continuando, já colocamos "imersaodev, alura" no campo das *tags*. Então temos a função Converter(), e agora criaremos a já conhecida função

`console.log()`. Quando clicarmos no botão de "Converter", queremos exibir uma mensagem com o texto "clicou" no console.

```
function Converter() {
  console.log("clicou")
}
```

Guilherme: Vamos ver se isso está realmente acontecendo. Salvaremos e rodaremos mais uma vez, aguardaremos o carregamento e não colocaremos nenhum valor, afinal ainda não pedimos ao programa que pegue este valor e realize as operações.

Clicando em "Converter", exibiremos a mensagem "clicou" no console, que era o que queríamos.

Rafaella: Exatamente. Sobre essa questão das funções, pegando como exemplo o `console.log()`, vemos que também tem um parênteses. Estamos mandando a informação, mas ainda não estamos fazendo isso nesta função pois faremos mais adiante nas próximas aulas. E

Então existem comandos que podemos utilizar que já são do JavaScript, como é o caso do `console.log()` - que tem todo um conjunto de código próprio para imprimir no console -, e existem outras que nós mesmos criamos e personalizamos com o que queremos que faça, como é o caso do `Converter()`.

Como ainda não estamos pegando as informações da tela, estamos apenas dizendo que, quando clicar no `onClick`, executará o `console.log()`. Começaremos a pegar esses dados da tela a partir agora.

Faremos isso olhando também o HTML, que pode parecer um pouco complexo de tentar entender todas as palavras no início, mas fica mais fácil com o tempo porque é bem intuitivo.

Por exemplo, é fácil entender que a tag `<img>` corresponde à imagem que está sendo exibida no projeto ao lado, então conseguimos localizar cada elemento.

O elemento que conseguimos entrar com os dados se chama input, e no código HTML conseguimos ver que o nosso `<input>` já está com um `id` chamado `"valor"` que é do tipo *number*, então irá receber `"number"` na propriedade `type`, afinal digitaremos um número no campo de texto que entra no elemento.

Da mesma forma que fazemos no CSS como dissemos anteriormente, colocamos o `<h1>` e depois o estilizamos. Estamos usando a própria tag para referenciar qual o elemento que queremos trocar.

O CSS vai estilizando as coisas do HTML, mas o JavaScript também fará desta forma referenciando os elementos do HTML de alguma maneira, e vai colocando dinamicidade na página.

Por exemplo, quando clicamos em um botão e algo acontece, isso não é feito no HTML e no CSS sozinhos, pois quem age de fato é o JavaScript.

Para isso, não usaremos a tag `<h1>` por exemplo, pegaremos exatamente o `id`. Mais adiante, quando formos utilizar mais HTML, veremos como é importante essa identificação para podermos referenciar o elemento que queremos no JavaScript.

De volta ao código JS, deixaremos o `console.log()` para alterarmos mais adiante, e começaremos criando uma variável na qual guardaremos o valor colocado no input em algum lugar da memória do computador através das variáveis.

Criaremos uma `var` com o nome `valorElemento` e, para pegarmos as informações do elemento HTML, existe um comando próprio do JavaScript usado chamado `document.getElementById()`, o qual é literalmente a tradução de "pegue o elemento por *id*".

Então passaremos o `id` que queremos dentro dos parênteses, e como neste caso é o valor do input, já sabemos que no HTML está como `"valor"`.

Portanto, colocaremos o valor do input do nosso HTML para a variável `valorElemento`, e agora poderemos colocá-la no lugar de `"clicou"` dentro dos parênteses do `console.log()` para conseguirmos ver o que pegamos do HTML.

Rodaremos o código, lembrando que isso acontece no HTML do botão `onClick` em que o `Converter()` está.

```
function Converter() {
  var valorElemento = document.getElementById("valor");
  console.log(valorElemento);
}
```

Rafaella: Clicando no botão "Converter", imprimiremos não apenas o valor daquele campo que adicionamos, e sim o código HTML inteiro do componente, mas não é o que queremos.

Guilherme: Estamos trazendo toda a estrutura HTML do `<input>`, porém queremos somente pegar o que foi digitado no campo, como o valor "10" por exemplo. Faremos isso em duas etapas para que fique bem claro.

Após valorElemento, criaremos uma nova variável chamada valor que será o valor do elemento, só que não o componente inteiro, e sim somente o conteúdo que está dentro, ou seja, o que foi digitado dentro da caixa.

Para isso, colocamos um ponto .value para acessar este conteúdo do valorElemento. Em seguida, exibiremos apenas este valor, salvaremos, limparemos o terminal e executaremos novamente.

```
function Converter() {
  var valorElemento = document.getElementById("valor");
  var valor = valorElemento.value;
  console.log(valor);
}
```

Guilherme: Na caixa de texto do projeto, escreveremos o valor "10" e clicaremos em "Converter". Com isso, aparecerá o valor "10" no console.

Rafaella: Não é exatamente o que queremos, pois quando estamos falando de valor - ainda não fizemos a conta da conversão de real para dólar ou vice-versa - mas lembrando que sempre que vemos algum valor entre aspas, na verdade estamos vendo uma *string,* ou seja, um valor em texto.

Então quando fazemos operações, é interessante utilizarmos o tipo real da variável. Como lidaremos com moedas, é interessante mudarmos para o tipo número ao invés de mantermos o textual.

Existem dois grandes tipos que usamos normalmente para tipos de números, que é o **inteiro**, ou seja, aquele que não possui vírgula, e o **decimal** que usa vírgula. Para cada um respectivamente, utilizamos int e float na programação.

Precisaremos fazer um comando específico, da mesma forma que fizemos no .toFix anteriormente, o qual fixava duas casas decimais. Agora queremos transformar em número o valor que foi inserido na caixinha como texto, afinal os valores digitados nos campos são impressos como textuais por padrão em JavaScript.

Este comando é chamado parseFloat(), e não usaremos parseInt() porque se trata de valores de moedas que possuem casas decimais após a vírgula. Com ele, transformaremos o valor de texto em número decimal.

Para isso, criaremos uma nova var em que colocaremos o valor em dólar, chamada valorEmDolarNumerico sendo igual a parseFloat(). Dentro dos parênteses, passaremos o valor textual que queremos transformar em numérico.

Neste caso, já temos a nossa própria variável `valor`, que é exatamente essa que tínhamos impresso no console como string. Depois, imprimiremos `valorEmDolarNumerico` no `console.log()` para vermos se de fato mudamos para número.

```
function Converter() {
  var valorElemento = document.getElementById("valor");
  var valor = valorElemento.value;
  var valorEmDolarNumerico = parseFloat(valor);
  console.log(valorEmDolarNumerico);
}
```

Guilherme: Colocaremos o valor "10" na caixa de texto novamente e clicaremos no botão para converter.

Feito isso, receberemos a impressão de `10` sem as aspas no console, indicando que realmente se trata de um número decimal.

Paulo: Recapitulando alguns pontos, toda vez que estamos terminando de alterar um código e queremos rodar de novo, apertamos a tecla "Enter" no código, depois clicamos em "Save" na barra superior de opções e por fim no botão "Run" para recarregar o documento e executar, sempre clicando em "Clear" para limpar o console e não nos confundirmos com o que estava antes.

Depois disso, inserimos algum valor na caixa de texto do projeto e clicamos no botão escrito "Converter". Quando fazemos isso, acontece o que chamamos de **evento** na página ao acionarmos o `onClick`, da mesma forma como vimos anteriormente quando passamos o cursor do *mouse* em cima de algo.

Então quando o botão é clicado, o programa busca a função `Converter()` que criamos, executando linha por linha do que contém. Agora tudo fica mais complexo, porque conseguiremos ver algumas coisas que são da *web,* são do navegador.

O `getElementoById()` já está na biblioteca do navegador, portanto o JavaScript já tem acesso para puxar o elemento do HTML com a identificação certa.

Muitos dos componentes visuais do HTML possuem `.value`, e conseguiremos pegar exatamente o que está dentro dele. E a função `parseFloat()`, por sua vez, já existe há bastante tempo no JavaScript.

Depois, chamamos de novo a outra função `console.log()` e outra que já havíamos visto na aula anterior, porém a cada linha estamos usando algo que já existe e dizemos ser da biblioteca, da API, é uma funcionalidade que já existe.

Isso pode assustar às vezes, pois há muitas coisas que não escrevemos e estamos chamando algo que alguém fez para que possamos utilizar, mas é muito comum em desenvolvimento, então ficamos mais confiantes e experientes com o tempo.

É sempre importante prestarmos atenção ao uso das letras minúsculas e maiúsculas na escrita destes comandos, pois se errarmos o nome, um erro é apresentado no console dizendo que não é uma função, e não conseguiremos executar o que precisamos.

Na dúvida, pergunte no Discord que iremos esclarecer qualquer questão que tenham, e a comunidade também pode ajudar. Mas ter certa dificuldade inicial em entender a linguagem e as mensagens em inglês é normal, então não se deixe abalar com os erros, tente entender o que está acontecendo e continue estudando, pois isso fará parte da vida do desenvolvimento.

Quando um erro aparece no console, o sistema indica onde está exatamente e dá indicação do problema para podermos resolver, então é bem cuidadoso. Podemos colocar mais uma dúvida? Vamos primeiro limpar o console.

Na aula anterior, estávamos vendo o código fora das funções, e queria colocar um novo `console.log()` depois das chaves da função `Converter()`, recebendo a mensagem `"passei por aqui"`.

Antes de salvarmos e rodarmos, quando será executado este segundo `console.log()`? Para quem está começando agora, esta pergunta não é trivial, e pode até gerar uma pequena confusão. Então vamos reparar no que irá acontecer.

```
function Converter() {
  var valorElemento = document.getElementById("valor");
  var valor = valorElemento.value;
  var valorEmDolarNumerico = parseFloat(valor);
  console.log(valorEmDolarNumerico);
}
console.log("passei por aqui")
```

Guilherme: Salvei. Vou executar, ele vai limpar.

Paulo: Antes mesmo de apertar o botão "Converter", ele executou a linha 7. Podemos nos perguntar o motivo de não ter executado a linha 2, 3, 4 ou 5? Na verdade, ele interpretou essas linhas, mas como ele sabe que tudo isso está dentro do `Converter()`, ele só vai, realmente, executar essas linhas quando alguém chamar, nós usamos o verbo chamar ou invocar o `Converter()`. Por enquanto, ninguém chamou. Funções são blocos de código que, em algum momento, podem ser chamados.

O que está fora da função, sim. O CodePen executará de cima para baixo tudo que estiver fora de função. Agora, o que está dentro da função, nós não sabemos quando

ele executará. Acontecerá quando e ela for chamada. Isso é outra coisa que começará a aparecer e, às vezes, gera um pouco de confusão. Mas, não desista, explore, pause o vídeo. Nós teremos exercícios para praticar, eles ficarão no extra.

Não deixe de testar, "E se eu colocar aqui? E ali? Se eu trocar a ordem, o que acontece?". Com isso, será possível concluir, tirar lógicas e compreender o funcionamento. Esse é um exercício muito importante. Mais uma vez, lembrando algo que a Rafaela falou nos primeiros minutos da primeira aula, é necessário ter curiosidade e criatividade.

Guilherme: Algo interessante, que você até comentou ontem, Rafa, é que demos ênfase ao **CamelCase**, sendo a primeira palavra minúscula e todas as outras começam com maiúscula. Isso é um padrão, uma convenção, do JavaScript. Nós mantemos isso para as nossas variáveis. Repare que estamos construindo, com o passar das aulas, um conhecimento consolidado. Estamos aperfeiçoando o que temos como ideia.

Rafaella: Vamos, agora, recapitular onde estávamos parados no código. Nós estamos transformando o valor que estamos passando no input em número do tipo *float*, número decimal. Nós estamos imprimindo-o no `console.log()`, logo abaixo, `valorEmDolarNumerico`. Porém, não estamos fazendo nenhuma conta e nós precisamos passar de dólar para real.

Se nós estamos com o dólar e, por exemplo, um Playstation 5 custa, não sei, na verdade, não tenho ideia de qual é o preço de um Playstation 5 em dólar.

Guilherme: Quinhentos dólares?

Paulo: Sim, eu diria quatrocentos dólares ou quinhentos. É muito dinheiro!

Rafaella: É muito dinheiro! Quinhentos dólares então. Para convertermos para o real, vamos supor que o dólar está a cinco reais, então, precisamos multiplicar por cinco para chegar na quantidade de reais que tem o dólar. Considerando quinhentos dólares vezes cinco, teremos dois mil e quinhentos reais. Então, devemos fazer essa conta no nosso código também. Já estamos com o `float`, temos que fazer o `valorEmDolarNumerico` vezes o número cinco e podemos colocar isso em uma variável.

Portanto, criamos uma variável nova `var valorEmReal =` e multiplicamos o valor em dólar numérico por cinco, ou seja, `var valorEmReal = valorEmDolarNumerico * 5`. Gui, vamos alterar no HTML onde está escrito "descubra os valores em Dolar U$", vamos colocar em real e já fica correto, "descubra os valores em real R$".

Guilherme: Está certo, na linha 6, teremos `"descubra os valores em real R$"`.

Paulo: Quando você, aluna e aluno, estiver vendo, já estará em real.

Rafaella: Sim, já estará. Isso é apenas para percebermos como é fácil encontrar as coisas no HTML e alterar.

Guilherme: Vamos verificar no console?

Rafaella: Sim, vamos dar um console.log(). Acho que podemos apagar o de cima, para não nos confundirmos.

```
function Converter() {

  var valorElemento = document.getElementById("valor");
  var valor = valorElemento.value;

  var valorEmDolarNumerico = parseFloat(valor);

  var valorEmReal = valorEmDolarNumerico * 5
  console.log(valorEmReal)

 }
```

Guilherme: Salvei, limpei o console e rodei. Vamos ver. Qual é o valor do Playstation 5 que vocês falaram?

Rafaella: Quinhentos dólares.

Guilherme: Na nossa página, aparece a frase "Conversor de moedas" e o subtítulo mudou para "Descubra os valores em real R$". Em seguida, está escrito "imersão dev_", "insira o valor", um campo para adicionar esse valor e um botão "converter". Ao inserir o valor "500" e apertar "converter", ele retorna o valor "2500". Deu certo.

Rafaella: Então, Gui, nós estamos imprimindo, entrando com os valores na tela, mas ainda estamos imprimindo no console e queremos que a pessoa veja o valor ser impresso em real na página. Como fazer isso? Colocar na tela esses 2500 reais, por exemplo?

Paulo: Se o console estiver fechado, nem veremos nada acontecendo. Vamos ficar apertando o botão "Converter" várias vezes e nada acontecerá.

Rafaella: Se você mandar para alguém, dificilmente a pessoa pensará em clicar com o botão direito, inspecionar, console, enfim, fazer todo esse processo. Precisamos imprimir na tela.

Guilherme: É verdade! Se pararmos para pensar na lógica usada, nós pegamos um elemento que estava no HTML com a palavra document e indicamos: pegue o documento, o HTML, e vá ao id valor. Nós faremos um processo muito semelhante

agora. Precisamos ir em algum lugar do HTML e dizer que queremos colocar o código JavaScript dentro dessa parte do HTML, porque temos que especificar onde mostraremos o 2500.

No HTML, as pessoas que desenvolveram foram muito generosas conosco e deixaram uma div: . Nela, não há nada.

Paulo: A **Div** é um espaço do HTML que podemos usar, uma divisão da nossa página web, é isso?

Guilherme: Isso! É isso mesmo.

Paulo: Queremos pegar esse elemento, mas, ao invés de pegar, desejamos alterar o valor dele.

Guilherme: Isso! Alterar o valor. O comando para realizarmos isso não é difícil. Poderia até dizer que ele é até um pouco semântico. Nós pegaremos e selecionaremos o , que está na linha 12. Para isso, utilizaremos outra propriedade, não o getElementById().

Paulo: Acho que eu tenho uma ideia. No console, fica ruim, certo, Rafa? Porque, de fato, ninguém ficará abrindo o console no celular para ver o resultado. Desejamos que apareça na página, pode ser logo abaixo do "Converter" ou em algum outro canto. Existem algumas opções de JavaScript, nós não estamos usando o alert() por vários motivos. A ideia é pegarmos outro elemento pelo id dele e, ao invés de pegar, trocaremos o valor. Isto é, trocaremos o HTML, o código que existe dentro.

Algo parecido com o var valorElemento, recebe o elemento que se chama valor, mas, ao invés de pegarmos o .value, vamos alterar o que tem dentro. Para isso, imagino que vocês já tenham deixado algo preparado, com algum lugar possível de ser alterado, certo?

Guilherme: Sim. No HTML, Paulo, existe um h2 que, as pessoas que desenvolveram esse Front, gentilmente, deixaram com um id="valor convertido"></h2. Nós faremos justamente o que você disse. Referenciaremos esse id e indicaremos que, agora, não queremos pegar um valor, mas, sim, colocar esse valor para exibirmos na tela e esse h2 ficará bem bonito exibindo o valor que convertemos.

Paulo: Então, nós pegaremos esse elemento de id="valor convertido".

Guilherme: Vou chamar de elemento para ficar claro o que estamos fazendo, var elementoValorConvertido = document.getElementById(), e pegaremos exatamente o id que existe no HTML, o valorConvertido.

Paulo: Até recomendo copiar e colar (Ctrl + C" e "Ctrl + V") para não arriscarmos errar.

Guilherme: É verdade, valorConvertido. Não se esqueçam do CamelCase! Colocaremos entre aspas, "valorConvertido", isto é, var elementoValorConvertido = document.getElementById("valorConvertido"). Pronto! Pegamos o valor. O que precisamos fazer agora é, de alguma forma, colocar o valorEmReal dentro desse valorConvertido. Faz sentido?

Rafaella: Certo! Eu acho que seria interessante não só pegar o valor, mas também colocar o resultado em real, por exemplo, para a pessoa não ficar só com "2500" sem nenhuma especificação. Poderíamos, portanto, criar uma variável para escrevermos esse texto, "O resultado em real é x". A nossa variável pode ser o valorConvertido e escreveremos esse texto, entre aspas, porque será do tipo *string*, texto, "O resultado em real é R$ " + valorEmReal".

```
var valorConvertido = "O resultado em real é R$ " + valorEmReal
```

Paulo: Então, pegamos o elemento e criamos o texto que deve aparecer. Agora, falta dizer que esse texto vai para dentro desse elemento. Deve existir outro comando para usarmos, certo? Porque, só isso, não basta. Pegamos o elemento e geramos a frase, mas ainda não reunimos as duas coisas.

Rafaella: Exato!

Guilherme: Então, vamos fazer. Nós já temos o elemento, nós precisamos indicar ao JavaScript que pegue o elemento, elementoValorConvertido, que é a nossa propriedade do HTML, e precisamos, de alguma maneira, dizer que queremos inserir nesse elemento o conteúdo da linha anterior, referente ao valor convertido e a mensagem. Para isso, utilizaremos o sinal de ponto final (.) e escreveremos "inner", com duas letras *n*, HTML, em letras maiúsculas, ou seja, elementoValorConvertido.innerHTML =. Existe até uma tradução do inner, não é, Rafa?

Rafaella: O significado de *inner* é "interior" ou "interno". Indica, portanto, que queremos colocar algo dentro desse elemento do HTML, entre as duas *tags*, de abertura e fechamento.

Guilherme: Nós queremos colocar o valorConvertido, que é o nosso texto, ou seja, elementoValorConvertido.innerHTML = valorConvertido. Vamos testar? Acho que não precisamos do console agora. Vamos salvar, rodar e colocar o valor do Playstation 5 e, quando apertamos "Converter", aparece "O resultado em real é R$ 2500".

Paulo: Parabéns! Ficou bem bacana! Se você chegou até aqui, isso é algo que você pode mostrar, que deve ter no seu portfolio, para atestar que você já entende um pouco sobre o JavaScript, sabe multiplicar, colocar variável, entre outros. Você pode até não se lembrar exatamente onde vai o float ou o .innerHTML. Vou te confessar uma coisa, eu também não sei. Eu não tenho a prática do dia a dia como a Rafa e o Guilherme tem, nessa linguagem, na web. Não é o meu domínio. E essa frequência é necessária.

Os dez dias conosco, essas semanas, são um começo, mas, você precisa continuar, precisa ter disciplina. Talvez, nao seja necessário estudar todos os dias, duas ou três horas, como nas nossas aulas, mas, você precisa continuar para saber o que está acontecendo no código e dominar, assim como a Rafa e o Guilherme dominam. Mesmo assim, em algum momento, eles podem se confundir também. Nesses momentos, basta testar, não ter medo, saber que não é esse o diferencial.

Claro que é muito bonito ver alguém que tem o domínio, sabe de cor toda a biblioteca, mas, realmente, isso não é essencial. Eu gostaria de colocar dois problemas que apareceram. Um deles, de propósito, não perguntei na hora que vi, porque aposto que, você que está assistindo, passou por esse erro e pensou "A Rafaella e o Guilherme nem falaram nada". Então, o primeiro é o seguinte: se eu digitar para converter a moeda e, ao invés de 500, eu escrevo "abacaxi" ou "playstation" no campo "Insira o valor"?

Vamos rodar de novo? Escreva para mim, Guilherme, "playstation".

Guilherme: Eu não consigo.

Paulo: No HTML, eles já usaram, no input, um `type="number"`. Se eles não tivessem colocado isso, nós poderíamos digitar qualquer coisa, não só números. Quando tentássemos tratar isso como número, no `parseFloat()`, teríamos um problema. Vou deixar isso como exercício para você testar. O problema de tentar converter uma palavra para número, que não era número, é muito comum. Vai estourar uma mensagem horrorosa na frente da tela. Esse era o primeiro ponto que eu gostaria de trazer.

O segundo, Guilherme, por favor, retorne ao código. Quando vocês estavam escrevendo, isso também apareceu, os vários sinais de ponto e vírgula (;) que as vezes você usou e, outras, vezes, não, tendo corrigido só depois. Se você deletar esses sinais de ponto e vírgula que existem ao final de cada instrução que estamos dando, algumas linguagens gostam, outras se confundem.

O JavaScript é uma das que, nesse caso, não tem problema algum, inclusive, muita gente diz para não usar, correto?

Rafaella: E o CSS, por exemplo, usa. Se você não coloca, vai dar errado.

Paulo: Então, temos todos esses detalhes de sintaxe, que demandam tempo para entender. Outro caso que também funcionaria, e que as vezes assusta é, por favor, Guilherme, aperte "Enter" no meio da linha, por exemplo, em `var valor =` e aperte "Enter" depois do igual. Se tivéssemos um ponto e vírgula no final, acredito que ele te aceitaria bem. Sem o ponto e vírgula, ele pensa que aquela parte já era uma linha.

Existem alguns detalhes que nem eu sei no JavaScript, mas, em algumas linguagens precisamos do ponto e vírgula e em outras nem existe. Esses detalhes da linguagem,

vocês conseguirão absorver com o tempo, quando dá erro e quando não, se podemos ou não fazer determinada coisa. Enfim, normalmente ele não gostará de ambiguidades, quando fica difícil compreender o que está acontecendo.

Guilherme: Um ponto legal, relacionado a isso que você comentou, Paulo, é: vou deixar meu código não *identado*. O **código identado** é o que não é escrito conforme a estrutura que nós temos. Observe que, na function Converter(), nós deixamos sempre um pequeno espaço para indicar que tudo que está na linha faz parte da nossa função Converter().

```
function Converter() {
  var valorElemento = document.getElementById("valor");

  var valor = valorElemento.value;
  var valorEmDolarNumerico = parseFloat(valor);

  var valorEmReal = valorEmDolarNumerico * 5
   console.log(valorEmReal)

          var elementoValorConvertido = document.getElementById("valorConvertido")
          var valorConvertido = "O resultado em real é R$ " + valorEmReal
          elementoValorConvertido.innerHTML = valorConvertido
 }
```

Eu vou tirar essa *identação* e colocar alguns trechos de código para cima, em outros vou dar um espaço e deixar o nosso código da seguinte forma:

```
function Converter() {
  var valorElemento = document.getElementById("valor");

  var valor = valorElemento.value;
  var valorEmDolarNumerico = parseFloat(valor);

  var valorEmReal = valorEmDolarNumerico * 5
  console.log(valorEmReal)
          var elementoValorConvertido =document.getElementById("valorConvertido")
          var valorConvertido = "O resultado em real é R$ " + valorEmReal
          elementoValorConvertido.innerHTML = valorConvertido
 }
```

Eu tirei todas as *identações* e, antes de salvar o projeto, vamos acessar "Settings", na próxima página, o "Behavior" tem uma última propriedade que se refere ao que o Paulo comentou: o ponto e vírgula. Eu mesmo não coloquei nenhum ponto e vírgula, ele foi

preenchendo sozinho depois, por causa do "Format on Save". Vou desativá-lo, salvar, e, em seguida, executar o projeto e vocês perceberão que ele continuará funcionando.

Salvei, rodei, e, observe que o código não está *identado*. Se eu colocar um valor para converter, funciona.

Paulo: Funciona, só é feio para quem está acostumado com outra organização, para quem vai ler e precisa compreender “o que está dentro do que”, nós costumamos dizer, o que está "aninhado" ao que. Fica um pouco difícil.

Guilherme: Exatamente! E é por isso que essa propriedade vem para ajudar, Paulo. Lembrem-se de uma coisa fundamental: nós nunca devemos escrever um código que só nós, sozinhos, entendemos. Se eu escrevo um código que a Rafa lê e não entende, o Paulo lê e não entende, enfim, se ninguém entende, não adianta nada, é um código que não terá uma manutenção fácil.

É sempre desejável escrever um código para ficar claro o que estamos fazendo e que outra pessoa que desenvolve e lê esse código consiga entender o que fizemos em cada linha. Por isso, para termos um código bonito, identado, nós precisamos acessar "Settings", apertar "Behavior" e colocar a propriedade "Format on Save" para sim, "On", desta maneira, todas as vezes em que salvarmos, ele vai modificar o código.

Só de ativar, ele já colocou ponto e vírgula em todos, sendo que eu não estava usando ponto e vírgula, e já *identou*, deixou a estrutura correta. Vamos supor que, no futuro não tão distante, tenhamos uma função e, dentro, outra função. Cada vez, vamos indo para o lado. Isso nos ajudará a conseguir visualizar bem a nossa estrutura, por exemplo: determinado trecho se refere a determinada função; esse outro, a esse trecho.

Então, é muito importante pensarmos não só em ter um código que funcione, mas, também, que seja legível. Sobre o nome de variável, por exemplo, `var valorElemento`. Nós poderíamos colocar o nome dessa variável de `v`? Poderíamos trocar `valorEmDolarNumerico` por `vN`? Sim,o projeto vai funcionar, mas será que está claro? Qualquer pessoa que ler isso compreenderá?

Por exemplo, a minha mãe não desenvolve, não trabalha com programação. Se ela ler `valorElemento` ou `valorEmDolarNumerico`, por mais que ela não programe, ela terá uma ideia do que seja isso. Este é um ponto muito importante, não basta o código estar funcionando, ele precisa de variáveis com nomes que façam sentido e, ao mesmo tempo, deixar o nosso código com uma estrutura correta.

Observe que, nas variáveis que eu criei, sempre existe um espaço antes da escrita que chamamos, na programação, de código *identado.*

Rafaella: Então, pessoal, nós fizemos o nosso conversor de moedas, entramos com o dólar e estamos imprimindo em real, para saber quanto custaria, de fato, um Playstation em real. Mas, temos alguns desafios para vocês fazerem. Eu vou começar dando o meu

desafio, que é: nós temos um botão "Converter", mas poderíamos adicionar um botão "Converter para real" e outro "Converter para Euro" ou outra moeda que você preferir.

Uma dica para como vocês podem fazer é procurar no HTML como é feito o botão, investigar no CSS como o botão é estilizado, para deixar um pouco parecido e lembrar do onClick também, considerando que a função teria que ser um pouco diferente. E vocês? Algum desafio para o pessoal?

Paulo: Eu quero que você faça um outro conversor para saber quantos anos demora para viajar na velocidade da luz de um lugar para uma outra estrela. Temos Sírius B e, dada essa estrela, digitamos quantos quilômetros ela está de distância da Terra e temos que saber quantos anos leva, na velocidade da luz, para chegar lá. Então, um conversor de quilômetros para anos-luz. Entenderam? É desafiador.

Guilherme: Pensando até no gatilho da aula passada, podemos fazer vários tipos de conversores. Partindo dessa ideia de conversor de euro, um botão para converter em euro, outro para converter em libra ou outra moeda, o que poderíamos pensar também é em um conversor de temperatura.

Ou seja, passamos uma temperatura, e temos o conversor da temperatura para fahrenheit, Celsius ou Kelvin, por exemplo. E teríamos, portanto, essas três opções. Seguindo a ideia da Rafa, que achei bem bacana.

Paulo: Vou deixar mais um. Posso? Esse mesmo conversor de moedas que, dado em dólar, aparece em real, também gostaria que, na linha abaixo, isto é, em outra linha, aparecesse o valor em *bitcoins*. Então, é outro desafio, porque você precisará mexer no HTML para ter outro h2, com outro id, ir ao código e fazer outra conta e alterar outro elemento.

Com isso, você repete o que já foi feito durante a aula de um modo um pouco diferente e esse é o caminho para sedimentar o conhecimento que está em ebulição na cabeça. Lembrando que o importante é entender o funcionamento do que aconteceu no código.

Escrevam a revisão passo a passo, gravem um vídeo falando, abram o canal, façam *streaming* dentro dos canais de áudio e que nós temos vários no Discord, e nós já estamos vendo vocês falando do projeto, apresentando, tirando dúvidas. Muita gente, simplesmente vai fazendo a aula e compartilhando lá, outras pessoas vão assistindo, é muito interessante. Então, até a próxima! Até amanhã! Foco, código, digitem, gastem o tempo e mostrem para nós. Tchau!

Rafaella: Até amanhã, pessoal!

Guilherme: Até mais!