Programação Estruturada

Aula 05 - Entendendo o seu Código: depuração

Videoaula 03: Conceitos de Depuração

 Videoaula 03: Conceitos de Depuração

Nesta videoaula, você conhecerá os principais conceitos de depuração, os quais estão presentes em todas as suas ferramentas. São eles:

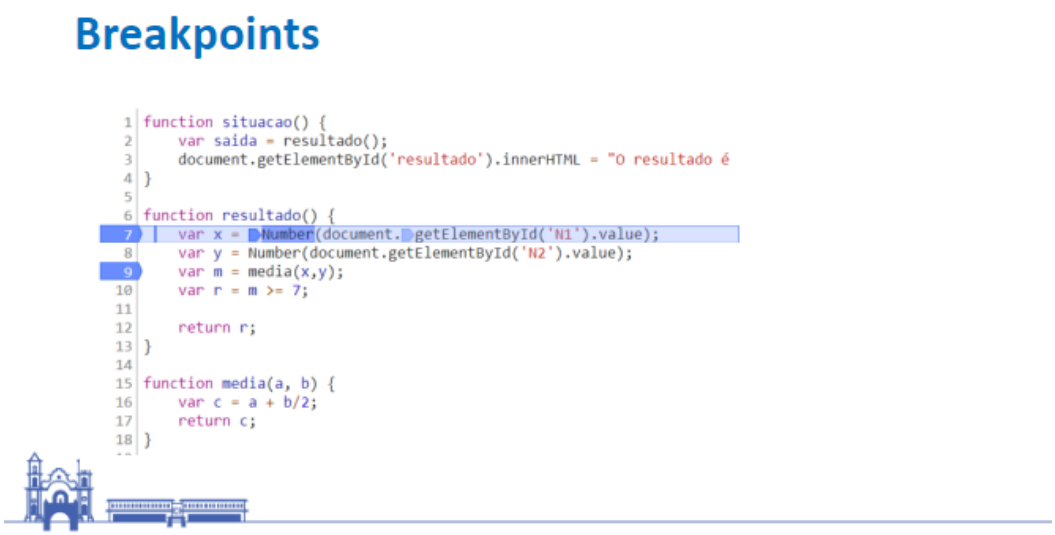
- Breakpoint
- Resume
- Step
- Step into
- Step out
- Step over
- Scope
- Watch List
- Call Stack

Conheça agora cada um deles mais detalhadamente.

O primeiro conceito é o de ***breakpoint***, ou seja, o ponto de parada. Ao executar um programa e ativar a ferramenta de depuração, o usuário poderá definir em que pontos ele deseja que a execução do programa pare para que possa analisar melhor os vários aspectos do programa naquele ponto, como, por exemplo, o valor das variáveis. Dentre os possíveis *breakpoints*, temos eventos DOM, *Global Listeners* e *Event Listeners*, os quais não serão vistos nesta disciplina. Além desses, podemos também definir *breakpoints* em linhas de código específicas, como você verá a seguir.

No slide, vemos um exemplo onde foi definido que o programa tem 2 *breakpoints*, nas linhas 7 e 9. Nesse caso, a execução do programa será pausada quando chegar na linha 7. Por exemplo, ao inserirmos os valores 2 e 3 e clicarmos no botão OK, a execução é pausada na linha 7.

Figura 1 - Execução pausada no Breakpoint da Linha 7



No slide, podemos ver que a ferramenta marca em azul a linha na qual a execução foi pausada. Neste momento, poderemos analisar o estado do programa naquele instante de execução. Após essa análise, para continuar com a execução a partir desse ponto, usamos um outro conceito, ***resume***, ou seja, prosseguir.

Essa operação continuará a execução do programa até o próximo *breakpoint*, caso ele exista, ou até o final do programa, caso contrário. Continuando com esse exemplo, se a execução estiver pausada na linha 7 e fizermos um *resume*, a execução continuará e será novamente pausada na linha 9, pois existe um *breakpoint* nesta linha, como podemos ver no slide.

Figura 2 - Execução pausada no Breakpoint da Linha 9

Note que ao final das linhas 7 e 8, as quais atribuem valores às variáveis `x` e `y`, temos uma indicação de que essas variáveis têm, de fato, os respectivos valores dados como entrada. Neste ponto, ao fazermos um novo *resume*, a execução acontecerá até o final, pois não existem mais *breakpoints*.

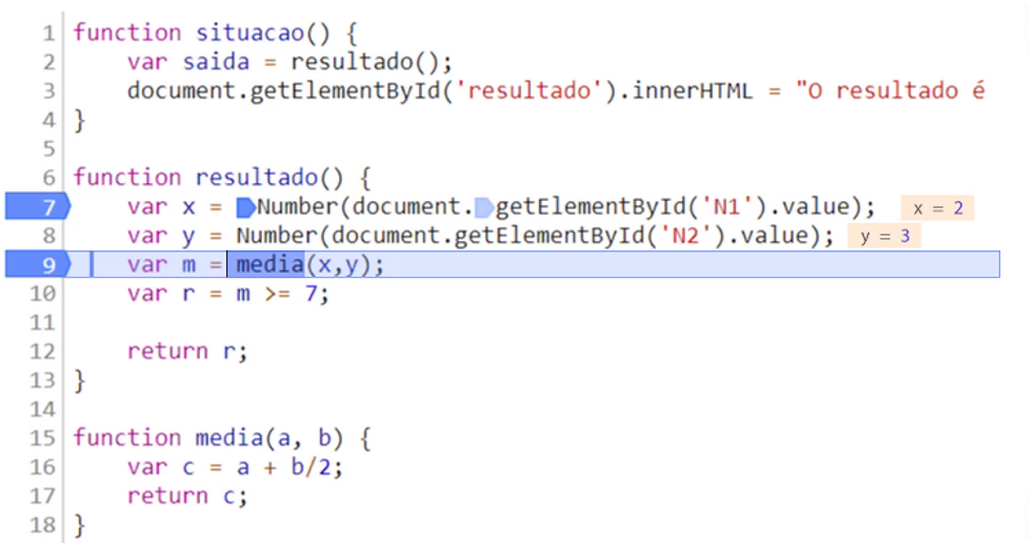
O próximo conceito, ***step***, corresponde a dar um passo na execução do programa. Na prática, essa operação executa a linha atual e pausa novamente a execução do programa.

Voltando ao exemplo, se a depuração estiver na linha 7, ao fazermos um *step*, podemos observar que agora a execução está na linha 8, pois a ferramenta está marcando essa linha.

Figura 3 - Execução pausada na Linha 8 após um Step

Neste momento, se novamente dermos um *step*, a linha 8 será executada e a execução pausada na linha 9.

Figura 4 - Execução pausada na Linha 9 após segundo Step



No entanto, se agora dermos um *step*, a execução irá para a linha 16, pois entramos na chamada da função `media` que tínhamos na linha 9.

Figura 5 - Execução pausada na Linha 16 após terceiro Step

Isso porque, no caso da ferramenta de depuração do Chrome, o *step* automaticamente entra nas funções definidas nesse script que estejam na linha executada. Ele tem o comportamento bastante similar ao ***step into***, que, como o nome diz (em inglês), entra na próxima chamada de função do script. Na verdade, no Google Chrome, a única diferença entre *step* e *step into*, é que o *step* não entra em funções assíncronas e o *step into* faz isso. No entanto, nesta disciplina, não abordaremos funções assíncronas e, portanto, ambas as operações terão o mesmo comportamento.

Uma vez dentro de uma função, a qualquer momento, usando o conceito de ***step out***, você pode optar por executar todo o resto do código da função em um só passo, retornando para o ponto seguinte à chamada da função. Por exemplo, se estivermos pausados na linha 16, como no slide, e dermos um *step out*, a ferramenta completará a execução da função `media` e pausará na linha 10.

Caso não deseje que a ferramenta entre em uma função, você poderá usar o conceito de ***step over***. Nesse caso, se a execução estiver pausada na linha 9, o *step over* executará a linha 9, não entrando na função `media`, já atribuindo o retorno dela à variável `m` e pausando na linha 10.

Figura 6 - Execução pausada na Linha 10 após um Step out

Por fim, temos os conceitos de escopo, lista de observação e pilha de chamadas, os quais se aplicam a determinados momentos da execução de um programa. O **escopo**, ou *scope*, em inglês, inclui todas as variáveis e funções que estão no escopo da linha na qual a execução está pausada. A **lista de observação**, ou *watch list*, em inglês, inclui expressões definidas pelo programador e apresenta o valor atual daquela expressão. Por fim, a **pilha de chamadas** de função, ou *call stack*, em inglês, é uma pilha que apresenta a ordem de chamadas de função que levou o programa até àquele ponto. A primeira chamada está na base da pilha e a última está no topo da pilha.

Por exemplo, no slide temos uma situação em que a execução está pausada na linha 16. Note que a pilha de chamadas tem a função `media` no topo da pilha e abaixo dela temos as funções `resultado`, `situacao` e `onclick`, nesta ordem. Isso porque a primeira chamada a qualquer função acontece no evento `onclick` do HTML, momento no qual chamamos a função `situacao`. Assim sendo, temos `onclick` na base da pilha e `situacao` acima dela. Depois disso, essa função chamou a função `resultado`, a qual é então colocada também na pilha. Por fim, a função `resultado` chama a função `media`, colocando-a também na pilha.

Figura 7 - Pilha de Chamada de Funções

Concluo assim a apresentação dos principais conceitos de depuração. Esses conceitos estão presentes em todas as ferramentas de depuração de programas, incluindo os depuradores dos principais navegadores de hoje. Na próxima videoaula, você verá um exemplo de uso desses conceitos de depuração no Chrome.