

TechOps – Testes para contratação

Abaixo estão alguns exercícios que visam validar conhecimentos gerais aplicados aos tipos de demandas mais comuns do dia a dia em TechOps.

As regras são simples:

- Todos os exercícios deverão ser respondidos usando somente HTML, CSS e JavaScript
- A entrega dos exercícios se dará através de um repositório Git seguindo a estrutura abaixo:

```
git/  
  -> natives/  
    -> ex1.html  
    -> ...  
  -> funis-de-compra/  
    -> ...  
  -> banners/  
    -> ...  
  -> logica  
    -> ...
```

Poderá utilizar qualquer serviço de Git de sua escolha, desde que o repositório seja acessível publicamente. Ao término dos exercícios, basta encaminhar a URL do repositório que devemos clonar para validarmos.

Natives

Natives são anúncios que, como o nome indica, possuem a estilização nativa do site em que serão exibidos. Em essência, são templates que no momento da impressão são preenchidos com os dados de campanha pelo Adserver.

VEJA TAMBÉM



Após saída de Douglas Tavalaro, CNN Brasil faz primeiro grande corte com nova direção



Exclusivo Acelerador Capilar Que Faz Crescer Até 15 Mil Fios Por Mês Chega ao Brasil.



Nova direção da Globo quer cortar vícios e implantar novo modelo de dramaturgia

Exercícios

1. Reproduza o layout da imagem abaixo utilizando HTML, CSS e JS, em um arquivo único, ou seja, escreva tudo dentro de "ex1.html". Preencha com dados falsos, ou seja, para os textos utilize "lorem ipsum"; para a imagem utilize alguma fonte de imagem randômica, como <https://picsum.photos/>



Cuidados de limpeza que todos os
“pais de pets” devem conhecer
para manter um lar feliz

Curiosidade: o elemento com fundo azul e texto “Cafuné” se chama *chapéu*

- Quando precisamos carregar documentos HTML dentro de outros documentos HTML, utilizamos [iframes](#). Reproduza o layout abaixo em um arquivo HTML, agora “ex2.html”, e preencha os 6 slots de anúncio carregando em cada um deles o arquivo desenvolvido no exercício anterior

Conteúdo da Web



Cuidar da casa e dos pets
ficou ainda mais fácil com
Cafuné
(Cafuné)



Cuidados de limpeza que
todos os “pais de pets” devem
conhecer para manter um ...
(Cafuné)



Entenda como alcançar o
equilíbrio perfeito de casa
limpa e pets saudáveis
(Cafuné)



Saiba mais do delivery da loja
de roupas Lez a Lez, você
pode provar os looks em ...
(LEZ A LEZ)



Saiba mais do delivery da loja
de roupas Lez a Lez, você
pode provar os looks em ...
(LEZ A LEZ)



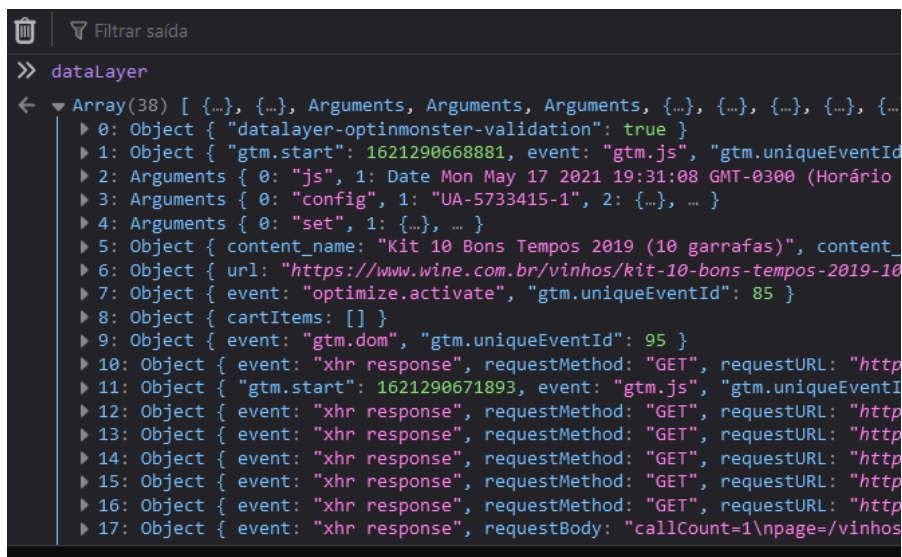
Adeus, bagunça: com os
cuidados certos, é possível
manter a casa e os pets ...
(Cafuné)

- BÔNUS:** utilizando JavaScript, adicione recursos de randomização nos natives, fazendo com que a cada carregamento eles tenham um visual diferente. Ideias: cores dos textos, camada semitransparente colorida por cima da imagem etc. Agora crie uma segunda vitrine de anúncios, como a do exercício anterior, para carregar os novos natives

Funis de compra

Funil de compra é o nome dado para a jornada do usuário dentro de um site de ecommerce. Normalmente é dividido em 5 etapas: home -> categoria -> produto -> carrinho -> checkout. A ideia é que muitas pessoas chegam até a home de uma loja; dessas uma porcentagem clica em alguma categoria de produtos de seu interesse; dessas uma porcentagem seleciona algum produto, que então podem colocar no carrinho; por fim, podem acabar comprando (checkout). Como a cada etapa que avança, menos pessoas chegam até ela, acabamos tendo um efeito de funil.

Um dos recursos mais importantes para podermos implementar toda a mecânica de marcação de etapas e usuários chama-se *dataLayer*. O *dataLayer* é um objeto que guarda diversas informações importantes em um ecommerce, armazenando dados e eventos utilizados por sistemas de métricas para análise de performance (comercial e técnica) do site, e que podemos, também, aproveitar para nossas marcações.



```
>> dataLayer
Array(38) [ {...}, {...}, Arguments, Arguments, Arguments, {...}, {...}, {...}, {...}, {...}
  ▶ 0: Object { "datalayer-optinmonster-validation": true }
  ▶ 1: Object { "gtm.start": 1621290668881, event: "gtm.js", "gtm.uniqueEventId": 85 }
  ▶ 2: Arguments { 0: "js", 1: Date Mon May 17 2021 19:31:08 GMT-0300 (Horário de Verão) }
  ▶ 3: Arguments { 0: "config", 1: "UA-5733415-1", 2: {...}, ... }
  ▶ 4: Arguments { 0: "set", 1: {...}, ... }
  ▶ 5: Object { content_name: "Kit 10 Bons Tempos 2019 (10 garrafas)", content_type: "product" }
  ▶ 6: Object { url: "https://www.wine.com.br/vinhos/kit-10-bons-tempos-2019-10-garrafas" }
  ▶ 7: Object { event: "optimize.activate", "gtm.uniqueEventId": 85 }
  ▶ 8: Object { cartItems: [] }
  ▶ 9: Object { event: "gtm.dom", "gtm.uniqueEventId": 95 }
  ▶ 10: Object { event: "xhr response", requestMethod: "GET", requestURL: "http://www.wine.com.br/vinhos/kit-10-bons-tempos-2019-10-garrafas" }
  ▶ 11: Object { "gtm.start": 1621290671893, event: "gtm.js", "gtm.uniqueEventId": 96 }
  ▶ 12: Object { event: "xhr response", requestMethod: "GET", requestURL: "http://www.wine.com.br/vinhos/kit-10-bons-tempos-2019-10-garrafas" }
  ▶ 13: Object { event: "xhr response", requestMethod: "GET", requestURL: "http://www.wine.com.br/vinhos/kit-10-bons-tempos-2019-10-garrafas" }
  ▶ 14: Object { event: "xhr response", requestMethod: "GET", requestURL: "http://www.wine.com.br/vinhos/kit-10-bons-tempos-2019-10-garrafas" }
  ▶ 15: Object { event: "xhr response", requestMethod: "GET", requestURL: "http://www.wine.com.br/vinhos/kit-10-bons-tempos-2019-10-garrafas" }
  ▶ 16: Object { event: "xhr response", requestMethod: "GET", requestURL: "http://www.wine.com.br/vinhos/kit-10-bons-tempos-2019-10-garrafas" }
  ▶ 17: Object { event: "xhr response", requestBody: "callCount=1\npage=/vinhos" }
```

Exercícios

Os exercícios abaixo devem ser feitos utilizando o site <https://www.americanas.com.br/>.

- Desenvolver dois códigos JavaScript (ou seja, “ex1a.js” e “ex1b.js”) para extrair os seguintes dados do *dataLayer*:
 - Etapa de “Categoria”
Imprimir no console duas informações:
 - Quantidade de Itens
 - Lista de IDs, separados por vírgula
 - Etapa de “Carrinho”
Imprimir no console duas informações:
 - Lista de IDs dos produtos selecionados, separados por vírgula
 - Valor total do carrinho
- Dependendo do site, podemos identificar a etapa do funil através de sua URL. Escreva um código que, analisando a URL do site, imprima no console se estamos na etapa “home”, “categoria” ou “produto”

3. BÔNUS: em um único código JavaScript, desenvolva um algoritmo que imprima no console as seguintes informações:

etapa atual: X

Ids listados: A, B, C, D, ...

Considere as etapas “home”, “categoria” e “produto”. Nas situações em que não exista ID para exibir, omitir a informação de “Ids listados”

Banners

Banners, em sua forma mais básica, são anúncios na Internet que visam atrair tráfego para o site do anunciante. Podemos pensar que são como “links ilustrados”.



Eles também formam a base para diversos outros formatos de anúncio, como os banners de Data Capture. Como o nome indica, esses anúncios têm como objetivo capturar os dados de pessoas interessadas pelo conteúdo ofertado. No caso de uma campanha de um novo imóvel, por exemplo, o anunciante (no caso, uma imobiliária) poderá montar uma base de pessoas interessadas pelo imóvel e posteriormente entrar em contato para tentar concluir a compra.



Exercício

As artes para os exercícios abaixo utilizam uma fonte proprietária da marca Heineken, que não será enviada junto. Utilize alguma fonte disponível em <https://fonts.google.com/> que seja o mais próximo possível

1. Dado o arquivo de referência “970x250-referencia.png” e utilizando os materiais fornecidos dentro da pasta “materiais/ex1/”, monte um banner animado no formato de 970x250. Esse banner não deverá ficar em loop, ou seja, a animação começa no ponto A (pouca coisa visível), vai até o B (todos os elementos visíveis) e para. Anime cada um dos elementos como achar melhor, limitando o tempo de animação total em 3s
2. **BÔNUS:** na pasta “materiais/ex2” há 3 arquivos de referência para o formato 300x250. Diferente do 970x250 feito no exercício anterior, por ser um formato menor não é possível exibir todos os elementos de uma só vez, então utiliza-se o artifício de transição entre diferentes steps. Utilizando os materiais fornecidos para o exercício anterior, você fará o que chamamos de “derivação”, ou seja, adaptará o que tem em mãos para o novo formato. Pode ser que não fique exatamente igual aos arquivos de referência, mas tente deixar o mais próximo possível. O banner deve ser animado, seguindo as orientações do exercício anterior

Para os exercícios abaixo pode-se utilizar quaisquer recursos do JavaScript aceitos pelos navegadores mais atuais. Como algumas especificações (especialmente as mais novas) podem ter pequenas diferenças de implementação em cada um dos navegadores, é preciso garantir que o código execute corretamente no Chrome e no Firefox.

1. Desenvolver um algoritmo que imprima uma sequência de número de 1 a 100 seguindo as regras abaixo
 - Caso o número seja divisível por 3, imprimir “Tech” ao invés do número
 - Caso o número seja divisível por 5, imprimir “Ops” ao invés do número
 - Caso o número seja divisível por 3 e 5, imprimir “TechOps” ao invés do número
 - Caso contrário, imprimir o próprio número
2. Transforme o código do exercício anterior em uma função que aceite como parâmetros “limit” e “steps”. Se “steps” for 0, imprimir erro.

```
function techOpsPrint(limit, steps) { ... }
```

- limit: a partir de 0, até qual número o algoritmo deverá rodar
- steps: de quanto em quanto deverá subir a sequência, de 2 em 2, de 3 em 3, etc.

3. Com base no exercício anterior, monte uma função que aceite os parâmetros “start”, “end” e “steps”. Agora a função poderá trabalhar com sequências tanto incrementais (de 1 à 100), como decrescentes (de 100 à 1), além de poder começar ou terminar em 0, que deve ser impresso como número.

```
function techOpsPrint(start, end, steps) { ... }
```

- start: número inicial da sequência
- end: número final da sequência
- steps: de quanto em quanto deverá subir ou descer a sequência, de 2 em 2, de 3 em 3, etc.

4. **BÔNUS:** reescreva o código do exercício 2 sem qualquer tipo de laço de repetição, ou seja, sem utilizar “for” ou “while”. Dica: recursividade
5. **BÔNUS:** reescreva o código anterior sem utilizar variáveis (var, let ou const) :P — ATENÇÃO: parâmetros de função não são considerados variáveis
6. Dada as duas listas abaixo, monte uma terceira contendo apenas os valores repetidos
 - Lista1: [13,21,40,8,37,15,39,41,19,17,3,4,48,27,38,24,47,31,45,10,5,29,11,46,49]
 - Lista2: [6,44,33,17,21,41,13,31,35,10,48,8,49,45,25,50,5,4,18,34,26,1,3,32,42]
7. **BÔNUS:** utilizando a seguinte URL https://www.wine.com.br/vinhos/tinto/cVINHOS-atTIPO_TINTO-p1.html monte dois algoritmos:

a) Com base no trecho de código **document.querySelectorAll(".js-productClick")** gere uma lista de elementos que possuam “title” como um de seus atributos e que a divisão do número no atributo “data-product-sku” por 3 não gere sobra, ou seja, “data-product-sku % 3 == 0”

b) Extraia do dataLayer uma lista com todas as entradas que possuam “event” valendo “xhr response” e que o campo “requestBody” não seja null ou undefined