

Introdução à Programação Web para Bioinformática

HTML, CSS, PHP & JavaScript

Diego Mariano
Raquel de Melo-Minardi



Laboratório de Bioinformática e Sistemas
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais

Introdução à Programação Web para Bioinformática: HTML, CSS, PHP & JavaScript

Primeira Edição

Diego César Batista Mariano, *M.Sc., Ph.D. student*
Profa. Raquel Cardoso de Melo-Minardi, *Ph.D.*

2017

Copyright ©2017. Todos os direitos reservados a Diego César Batista Mariano.

Mariano, D. C. B.; de Melo-Minardi, R. C.;

Introdução à Programação Web para Bioinformática: HTML, CSS PHP & JavaScript - 1ª edição

Belo Horizonte, 2017

KDP, Amazon, Brasil

ISBN: 9781520895154

Selo editorial: *Independently published*

1. Bioinformática 2. Introdução à programação 3. PHP 4. JavaScript 5. jQuery

Os autores incentivam a livre divulgação do material contido neste livro, assim como a reprodução por fotocópia ou distribuição por meio eletrônico, desde que sejam respeitados os termos estabelecidos pela Amazon.

Agradecimentos

Os autores gostariam de agradecer às agências de fomento à pesquisa Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG) e Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

CAPES / Edital Biologia Computacional / número de processo 23038.004007/2014-82

Agradeço também:

Aos primeiros que leram esta obra e de certa forma influenciaram em seu resultado final;

À do Equipe da InfoVis, que inspirou este trabalho;

A Carol Almeida Fotografia;

A Denis Guilherme;

A José Renato.

Sumário

NOTA DO AUTOR **19**

MATERIAL SUPLEMENTAR **21**

PREFÁCIO **25**

UMA BREVE HISTÓRIA DA INTERNET 25
POR QUE WEBSITES SÃO IMPORTANTES PARA BIOINFORMÁTICA? 27
COMO COLOCAR UM WEBSITE NA INTERNET? 28
VERSÕES DE UM SITE 32
O DOGMA CENTRAL DO DESENVOLVIMENTO WEB 33
A QUEM SE DESTINA ESTE LIVRO? 34
COMO ESTE LIVRO ESTÁ ORGANIZADO? 35

1. INTRODUÇÃO AO HTML **41**

1.1 HISTÓRIA 42
1.2 HTML5 42
1.3 SINTAXE HTML 43

2. FUNDAMENTOS DO HTML **45**

2.1 ELEMENTOS DE UMA PÁGINA HTML 45
2.1.1 Cabeçalho <head> *48*
2.1.2 Corpo <body> *49*
2.2 COMENTÁRIOS EM HTML 49
2.3 FORMATAÇÃO DE TEXTOS 51
2.4 FONTES 53
2.5 APLICANDO CORES 55
2.6 QUEBRAS DE LINHA
 58
2.7 MARCAÇÕES DE ÊNFASE 59
2.8 ALINHANDO TEXTO 61
2.9 TÍTULOS 62

2.10 RÉGUA HORIZONTAL E LISTAS 63
2.10.1 Régua horizontal <hr> *63*
2.10.2 Listas *65*
2.11 IMAGENS 67
2.11.1 Inserindo imagens *68*
2.11.2 Atributos da tag <img> *69*
2.12 LINKS 70
2.12.1 Ligando páginas *71*
2.12.2 Atributos da tag <a> *74*
2.13 TABELAS 75
2.13.1 Mesclando colunas *76*
2.13.2 Mesclando linhas *76*
12.13.3 Atributos para construção de tabelas *77*
2.13.4 Outras tags usadas na construção de tabelas *78*
2.13.5 Construindo uma tabela completa com HTML5 ..... *78*
2.14 FORMULÁRIOS 80
2.14.1 Rótulos <label> *82*
2.14.2 Tag <input> *83*
2.14.3 Textarea *89*
2.14.4 Caixas de seleção *90*

3. ESTRUTURA DE PÁGINAS 93
3.1 DIVISÓRIAS <DIV> 94
3.1.1 Definindo o tamanho de divs *95*
3.1.2 Posicionando divs *96*
3.1.3 Propriedade float *99*
3.1.4 Margens *102*
3.1.5 Div container *104*
3.2 CABEÇALHO <HEADER> E RODAPÉ <FOOTER> 106
3.3 BARRA DE NAVEGAÇÃO <NAV> E MENU <MENU> 108

3.4 ARTIGOS <ARTICLE> E SEÇÕES <SECTION> 109
3.5 BARRA LATERAL <ASIDE> 110

4. FOLHAS DE ESTILO EM CASCATA (CSS) 111
4.1 POR QUE CASCATA? 112
4.2 FOLHAS DE ESTILO DO DESENVOLVEDOR 113
4.3 SINTAXE 115
4.4 SELETORES 117
4.4.1 Classes (class) e identificadores (id) 117
4.4.2 Tipos de seletores 119
4.4.3 Seletor universal 120
4.5 PRINCIPAIS PROPRIEDADES 120
4.5.1 Propriedades para textos 120
4.5.2 Tamanho e margem 122
4.5.3 Fundo 125
4.6 CSS3 126
4.6.1 Gradiente 127
4.6.2 Bordas arredondadas 129
4.6.3 Sombras 130
4.6.4 Transparências 132

5. BOOTSTRAP 135
5.1 COMO INSTALAR O BOOTSTRAP? 136
5.2 COMEÇANDO A USAR O BOOSTRAP 136
5.3 SISTEMA DE GRID 138
5.4 TABELAS BOOTSTRAP 147
5.4.1 Bordas 150
5.4.2 Linhas intercaladas por listras 152
5.4.3 Linha destacada 153
5.4.4 Tabela condensada 154

5.5 FORMULÁRIOS 155
5.5.1 Formulário em linha *156*
5.5.2 Desativando campos *157*
5.5.3 Largura de campos *158*
5.5.4 Altura de campos *160*
5.5.4 Grupos de inputs *161*
5.6 BOTÕES 163
5.6.1 Cores para botões *164*
5.6.2 Tamanhos de botões *166*
5.6.3 Grupos de botões *167*
5.6.4 Tamanhos de grupos de botões *168*
5.7 IMAGENS 170
5.7.1 Imagens responsivas *171*
5.7.2 Formas de adornos *171*
5.8 ÍCONES 172
5.8.1 Biblioteca de ícones Bootstrap *172*
5.9 BARRA DE NAVEGAÇÃO 174
5.9.1 Barra de navegação fixa *177*
5.9.2 Barra de navegação inversa *178*
5.10 ELEMENTOS INFORMATIVOS 179
5.10.1 Rótulos *179*
5.10.2 Alertas *180*
5.10.3 Painéis *181*
5.10.4 Barras de progresso *182*
5.11 MAIS INFORMAÇÕES 184

6. INICIANDO A CONSTRUÇÃO DE UM WEBSITE 185
6.1 OBJETIVOS DO WEBSITE 185
6.1.1 O que nosso site modelo precisa ter? *186*
6.2 WIREFRAME 187

6.3 MOCKUP 188
6.4 PROTÓTIPO 194

7. INTRODUÇÃO AO PHP 211
7.1 HISTÓRIA 212
7.2 PHP 7 213
7.3 COMO PHP FUNCIONA? 213
7.4 INSTALANDO PHP 214
7.4.1 MacOS 214
7.4.2 Instalando XAMPP no Windows 10 215
7.4.3 Instalando Apache no Ubuntu 223
7.5 SERVIDOR PHP 224
7.6 *HELLO WORLD*, PHP! 225
7.6.1 Comandos echo e print 228

8. FUNDAMENTOS DO PHP 229
8.1 COMENTÁRIOS 229
8.2 VARIÁVEIS E TIPOS DE DADOS 230
8.3 INDENTAÇÃO DE CÓDIGO 232
8.4 OPERAÇÕES MATEMÁTICAS 234
8.4.1 Operadores de reconhecimento 238
8.4.2 Determinando se um número é par ou ímpar 239
8.4.3 Incremento e decremento 240
8.5 OPERADORES RELACIONAIS E LÓGICOS 242
8.5.1 Operadores relacionais 242
8.5.2 Operadores lógicos 243
8.6 ESTRUTURAS DE CONTROLE 244
8.6.1 Comandos condicionais 244
8.6.2 Organização de condições com parênteses 249
8.7 LAÇOS DE REPETIÇÃO 250

8.7.1 While *251*
8.7.2 Do... while *252*
8.7.3 For *253*
8.7.4 Foreach *253*
8.7.5 Cláusula continue e break *254*
8.7.6 Qual o melhor laço de repetição? *257*
8.8 FUNÇÕES 258
8.8.1 Retornando valores *259*
8.8.2 Chamando funções dentro de outras funções *260*
8.9 *STRINGS* 262
8.9.1 Diferença entre aspas e apóstrofos *262*
8.9.2 Concatenação de strings *263*
8.9.3 Convertendo letras maiúsculas e minúsculas *265*
8.9.4 Obtendo o tamanho de um texto *265*
8.9.5 Buscando substrings em uma string *266*
8.9.6 Dividindo strings com a função substr *267*
8.10 ARRAYS 270
8.10.1 Acessando elementos individualmente dentro de um array *272*
8.10.2 Inserindo e removendo elementos em um array *274*
8.10.3 Contando o número de elementos em um array *276*
8.10.4 Ordenando arrays *277*
8.10.5 Convertendo strings em arrays e vice-versa *279*
8.11 ARQUIVOS 280
8.11.1 Abrindo arquivos *280*
8.11.2 Fechando arquivos *282*
8.11.3 Lendo o conteúdo de arquivos *283*
8.11.4 Lendo arquivos como arrays *284*
8.11.5 Gravando em arquivos *284*

8.11.6 Lendo todos os arquivos de uma pasta.....285
8.12 FUNÇÃO INCLUDE.....288
8.13 FUNÇÃO SYSTEM 289
8.14 PRINCÍPIOS DA ORIENTAÇÃO A OBJETOS 292

9. TRANSFORMANDO UM WEBSITE EM DINÂMICO 297
9.1 CRIANDO O INDEX.PHP.....297
9.2 COMPREENDENDO A MODULARIZAÇÃO 303
9.3 CRIANDO A PÁGINA "SOBRE".....304
9.4 CRIANDO O PÁGINA NAVEGAR.....307
9.4.1 Criando um banco de dados sem um SGDB.....307
9.4.2 Criando o arquivo navegar.php.....308
9.5 CRIANDO A PÁGINA DE BUSCAS COM BLAST 315
9.5.1 Fazendo download do BLAST.....316
9.5.2 Construindo um formulário de buscas 316
9.5.3 Processando os resultados da busca 318

10. INTRODUÇÃO AO JAVASCRIPT 325
10.1 HISTÓRIA.....326
10.2 DECLARANDO UM *SCRIPT* COM JAVASCRIPT.....327
10.3 OLÁ MUNDO, JAVASCRIPT!.....328
10.4 PEGANDO DATA E HORA COM JAVASCRIPT 330

11. FUNDAMENTOS DO JAVASCRIPT 333
11.1 COMENTÁRIOS.....333
11. 2 VARIÁVEIS 334
11.3 OPERADORES MATEMÁTICOS 335
11.4 CONSOLE.....337
11.5 ESTRUTURAS DE CONTROLE 339
11.5.1 Comandos condicionais.....341
11.5.2 Loop while.....343

11.5.3 Loop for *344*
11.6 *ARRAYS* E OBJETOS 345
11.7 FUNÇÕES 348
11.8 EVENTOS 349
11.9 DOM 357
11.9.1 Métodos *358*
11.9.2 Adicionando ou removendo elementos *359*
11.9.3 Eventos e DOM *360*
11.9.4 Alterando o CSS *360*
11.10 FUNÇÃO *RANDOM* 364

12. JQUERY 367
12.1 HELLO WORLD, JQUERY! 367
12.2 ESCONDENDO UM ELEMENTO 370
12.3 MUDANDO CSS 372
12.4 LENDO ARQUIVOS LOCAIS COM JQUERY 375

13. BOOTSTRAP JAVASCRIPT 379
13.1 DECLARANDO O BOOTSTRAP JAVASCRIPT 379
13.2 MENU SUSPENSO 380
13.3 POPOVER 382
13.4 MODAL 384
13.4.1 Botão de chamada para o modal *385*
13.4.2 Divs do modal *386*
13.4.3 Largura de modais *389*
13.5 CARROSSEL 390

14. D3JS 397
14.1 GRÁFICO DE PIZZA (*PIE CHART*) 397
14.2 GRÁFICO DE BARRAS 402

15. 3DMOL 413

15.1 REQUISITOS BÁSICOS....414
15.2 VISUALIZANDO ESTRUTURAS DE PROTEÍNAS....415
15.3 GERANDO MÚLTIPLAS VISUALIZAÇÕES....418
15.4 DESENVOLVENDO COM 3DMOL....419

16. PROJETO FINAL....423
16.1 INSERINDO OS TEXTOS INICIAIS....424
16.2 PÁGINA DE AJUDA....425
16.3 MELHORANDO A PÁGINA NAVEGAR COM JAVASCRIPT....427
16.3.1 Buscando nome do organismo....429
16.3.2 Visualizando proteínas....431
16.3.3 Tabela Navegar....439
16.3.3.1 Preenchendo a tabela....440
16.4 MELHORANDO OS RESULTADOS DO BLAST COM JAVASCRIPT445
16.4.1 HTML DA PÁGINA BLAST (BUSCAR)....445
16.4.2 Processando a requisição do BLAST....448

EPÍLOGO....457

REFERÊNCIAS BIBLIOGRÁFICAS....459
Diego Mariano....461
Raquel Minardi....462

Nota do autor

Olá mundo! Após mais de dois anos de trabalho, chegamos ao terceiro dos três livros planejados inicialmente para esta série. Antes de começarmos a falar deste livro, seria interessante relembrarmos um pouco do que já foi apresentado. Anteriormente aprendemos Python e sua robusta biblioteca Biopython. A seguir vimos Perl, uma linguagem poderosa para processamento de texto. Sempre imaginei a série de livros "Introdução à Programação para Bioinformática" como uma trilogia, que o povo nas ruas já chama até de "A trilogia Bioinfo" (está bem, eu inventei essa parte). Então, falar de linguagens para Web talvez seja a melhor maneira de encerrar a série.

Neste livro aprenderemos a criar páginas para Internet, ou o que costumamos chamar de Websites. Páginas da Internet podem ser importantes para divulgação científica ou até mesmo para permitir múltiplo acesso a um *software*, o que podemos chamar de ferramentas Web.

Entretanto falar de Web envolve muitos conceitos. Teremos que compreender como a Internet funciona e aprender suas formas de comunicação. Também precisaremos aprender as linguagens de programação para Web, tanto as que são executadas no lado do servidor, chamadas de *server-side*, quanto as que são executadas no cliente, chamadas de *client-side*. Mas o que são servidores e o que são clientes? Por termos que explicar tantos termos, esse livro será um pouco maior que os anteriores. Mas não se assuste! Os conceitos serão apresentados de maneira bem compreensível.

Agora vamos aos anúncios válidos para qualquer livro desta série. Este livro adota uma linguagem coloquial. Garanto que isso foi feito a fim de buscar uma maior interação com nossos leitores. Declaro que todos os *scripts* apresentados neste livro foram testados nos sistemas operacionais Windows 10, Linux Ubuntu 16.04 e MacOS X. Entretanto, falhas estão sujeitas a acontecer (Lei de Murphy), como as detectadas no primeiro e segundo livro da série. Caso obtenha problemas ao executar qualquer *script*, procure utilizar as versões sugeridas.

Bom se você está gostando da série, tem alguma sugestão ou crítica, ou só queira dar um "oi" peço que entre em contato comigo pelo Twitter. É o método mais rápido para que eu responda. Envie um *tweet* para @diiegomariano (não está escrito errado, "diiego" é escrito desta forma mesmo).

Pronto para último passo na jornada da programação na bioinformática? Boa leitura!

Diego Mariano

http://diegomariano.com

Material suplementar

Este livro contém a mais surpreendente e fantástica coleção de materiais suplementares. Sei que sou um pouco exagerado, mas dessa vez apresento uma vasta quantidade de material suplementar.

Sempre que encontrar este símbolo:



Ele o levará para a código-fonte do *script* em questão, disponível no GitHub. Todos os *scripts* e exemplos preparados neste livro foram disponibilizados em: <https://github.com/dcbmariano/web>. Antes de começar a leitura do próximo capítulo, acesse o diretório e faça *download* dos dados contidos nesse repositório (clique em *Download Zip*).

Padrão de nomeação dos *scripts*:

- cX_sY.html
- cX_sY.php
- cX_sY.js

Como por exemplo, no *script* c1_s1.php:

- **c1** indica que o *script* é exibido no capítulo 1;
- **s1** indica que se trata do primeiro *script* exibido no capítulo.

Sempre que encontrar este símbolo:



Indica que o conteúdo apresentado possui uma vídeo-aula que o sumariza.

#	Vídeo-aula	Tempo	Link
1	Apresentação	00:50	https://youtu.be/jO-74s31Pik
2	Bioinformática	15:34	https://youtu.be/R-HcjHDLbRE
3	Por que programar na web para Bioinformática?	00:48	https://youtu.be/60SDdLRCv3o
4	HTML	00:32	https://youtu.be/RV7Z2MC8DNI
5	Fundamentos do HTML	06:14	https://youtu.be/S-TpzDIGIuY
6	TAGS	05:15	https://youtu.be/P-KyydwvjT4
7	Imagens	03:05	https://youtu.be/u6xV3oMxFs0
8	Links	03:36	https://youtu.be/gegpChmS_lA
9	Formulários	10:50	https://youtu.be/Vmk49Qgwk3o
10	CSS	01:01	https://youtu.be/40zy_hiWrcI
11	Fundamentos da CSS	12:24	https://youtu.be/o6CU2tyKS_c
12	CSS3	08:02	https://youtu.be/_mjbJjTDQ-M

13	Bootstrap	01:06	https://youtu.be/WSUtEcb8GHY
14	Fundamentos do Bootstrap	07:29	https://youtu.be/rLAXx6KjPq4
15	Sistema de grid (Bootstrap)	08:17	https://youtu.be/bL_BVuWqckE
16	Tabelas (Bootstrap)	04:47	https://youtu.be/WRoGEX8kChk
17	Botões (Bootstrap)	02:14	https://youtu.be/P_G_MidrL6c
18	Adornos para imagens (Bootstrap)	01:28	https://youtu.be/poPD8m-EfIw
19	Ícones (Bootstrap)	01:32	https://youtu.be/j5GkJMR-vZc
20	PHP	01:41	https://youtu.be/6koVjayMysU
21	Introdução ao PHP	22:08	https://youtu.be/qcXLcx--a7I
22	Como instalar PHP?	05:49	https://youtu.be/ejEXbY6-sdc
23	Variáveis em PHP	01:31	https://youtu.be/YOTQA20wmtQ
24	Operações em PHP	01:13	https://youtu.be/1s4lt_DS8s8
25	Comandos condicionais em PHP	02:37	https://youtu.be/6q2L7aps32Q
26	Laços de repetição em PHP	08:04	https://youtu.be/9l7HY6GeDbI
27	Envio de arquivos com PHP	06:54	https://youtu.be/ir5Xo914QMc
28	JavaScript	01:15	https://youtu.be/wPED37E0Ofw
29	Introdução ao JavaScript	04:03	https://youtu.be/g5-ej4TWolU
30	Fundamentos do JavaScript	03:25	https://youtu.be/sYR_iZ6ybHA

31	JSON	05:34	https://youtu.be/KXT1hyVk0p4
32	Modal (Bootstrap)	10:23	https://youtu.be/E0AYPAR6edQ
33	Carousel (Bootstrap)	08:41	https://youtu.be/uu14hIqW2LM
34	3Dmol	07:29	https://youtu.be/2wtd5FgJ5NM
35	Criando uma página HTML	19:52	https://youtu.be/Zr3HNcvgFcc
35	Criando páginas dinâmicas	08:02	https://youtu.be/AAyO-S7WpNw

Dica

Faça o *download* do aplicativo **ENTiTi** para Android/IOS e busque pela expressão "intro_web_bioinfo", aponte a câmera do seu *smartphone* para a capa do livro e veja a apresentação do livro em **realidade aumentada**.

Prefácio



1. Apresentação

https://youtu.be/jO-74s31Pik

Uma breve história da Internet

No princípio havia apenas alguns computadores. Eram grandes, estrondosos, barulhentos. Alguns ocupavam prédios inteiros. Realizavam poucos cálculos simples. Basicamente cálculos de balística de mísseis e decodificação de mensagens simples. Grande parte deles nada armazenava.

Com o tempo foram se tornando menores. Mais potentes. Nem um milionésimo da potência dos que existem hoje, mas mesmo naquela época já eram vitais. Processavam dados vitais. Armazenavam dados vitais. E um dos primeiros a verem isso foi o Departamento de Defesa dos Estados Unidos. Mas de que adiantava todo esse potencial, se as máquinas não pudessem se comunicar? Então criaram um meio de interligá-las. Perceberam que não poderiam concentrá-las em um único ponto. Seria uma vulnerabilidade. Era 1969, auge da guerra fria. Então, grandes cabos que passavam debaixo da terra, chamados *backbones*, interligaram as principais bases militares e centros de pesquisa dos Estados Unidos. Não havia um ponto central. Se qualquer base fosse atacada, outras

poderiam continuar operacionais. Surgia assim, a ARPANET, a primeira rede remota entre computadores.

Anos depois, universidades obtiveram permissão para acessar a ARPANET. Além disso, com a adoção do protocolo TCP/IP, a ARPANET começou a crescer exponencialmente. Assim, ela foi dividida em MILNET, rede usada para assuntos militares, e INTERNET, rede de acesso público.

Mais tarde, Tim Berners-Lee, um cientista da computação e professor do MIT, apresentou o protocolo HTTP, que permitia a comunicação por meio da internet entre um cliente, que gerava solicitações, e um servidor, que processa e retornava novas informações. Berners-Lee também criou o HTML, que na época era apenas um conjunto de ferramentas para facilitar comunicação entre grupos de pesquisa. Em 1991, ele criou uma página que apresentava a chamada *World Wide Web* (WWW), ensinou como acessá-la e como disponibilizar dados. A *World Wide Web* permitia que uma página de hipertexto armazenada um computador remoto, chamado de servidor, poderia ser acessada por outros computadores, chamados de clientes, desde que esses obtivessem acesso a ela por meio de um endereço e de um *software*, chamado de navegador (*browser*). Esse foi o grande marco para popularização da Internet. Nos anos 2000, o acesso a Internet estourou. Hoje quase não é possível imaginar a vida sem ela.



2. Bioinformática

https://youtu.be/R-HcjHDLbRE

Por que Websites são importantes para Bioinformática?

Mas você deve estar se perguntando onde entra a Bioinformática nisso? Como discutimos nos livros anteriores, a Bioinformática é uma área de pesquisa que possui um braço na Ciência da Computação e outro nas Ciências Biológicas. Sua principal função é permitir que um grande volume de dados pudesse ser processado, fazendo com que padrões humanamente indetectáveis pudessem ser descobertos. Isso poderia, em tese, auxiliar na explicação de fenômenos biológicos, na descoberta de novos remédios, na descoberta do mecanismo de ação de doenças, o que poderia trazer novas curas.

Entretanto, para tudo isso seria necessário analisar as tais grandes bases de dados, àquelas mesmas obtidas pelos tais equipamentos de próxima-geração. Mas onde elas estão armazenadas? Em grandes centros de pesquisa é claro! E, como você, um simples aprendiz de pesquisador, talvez até mesmo um grande pesquisador, mas sem as credenciais ou verbas necessárias para encontrá-las fisicamente, poderia ter acesso a elas? Pela internet é claro!

A internet permite que qualquer pessoa do mundo possa ter acesso a grandes resultados de pesquisas, possa coletar os dados usados na

pesquisa e reproduzi-la em sua própria casa (se os métodos forem bem redigidos é claro). A internet também permite que você construa seus próprios programas, faça suas análises e compartilhe seus resultados com o mundo. Assim, a internet e suas linguagens de construção de Websites se tornaram importantes ferramentas de divulgação científica.



3. Por que programar para bioinformática?

https://youtu.be/60SDdLRCv3o

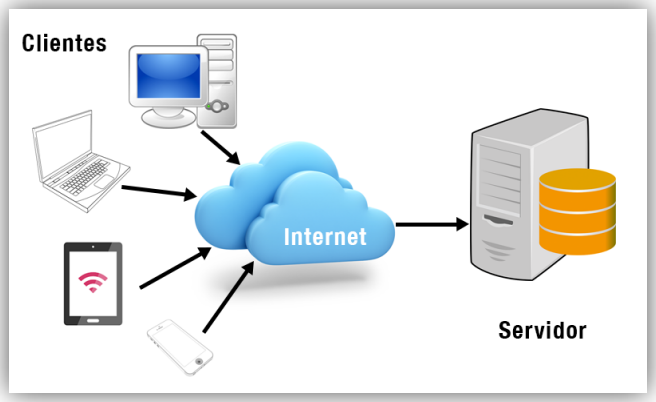
As primeiras ferramentas para bioinformática foram construídas para serem executadas localmente. Ou seja, para usá-las, era necessário possuir o código-fonte e compreender como executá-lo. Só quando essas ferramentas passaram a ser hospedadas em grandes servidores é que todos realmente tiveram acesso simples e fácil. Várias ferramentas poderosas estão disponíveis na internet como por exemplo a ferramenta de alinhamento local de sequências, BLAST: https://blast.ncbi.nlm.nih.gov/Blast.cgi.

Como colocar um Website na Internet?

Aqui apresentaremos o chamado paradigma cliente-servidor. Creio que todos nós já ouvimos falar que a internet é como uma nuvem. Você se conecta a nuvem e tem acesso a todos os dados.

A internet é uma rede de computadores interligados por cabos e outras coisas (existem várias tecnologias para transmissão de dados, entretanto convém dizer que conexões cabeais são mais rápidas e confiáveis). Esses cabos levam às operadoras que possuem os chamados roteadores, equipamentos que definirão as rotas para acessar os computadores que possuem as informações que você necessita. Quanto aos chamados *backbones*, pense neles como rodovias que interligam a internet a outros países (há gigantescos cabos que passam no fundo do mar conectando todo o mundo).

Enfim, é importante dizer que um Website (ou simplesmente site) que você deseja acessar está em um computador em outra parte do mundo. Esse computador deve estar ligado o tempo todo, ou então, como você poderia acessar o site a hora que quisesse? Logo, pode-se dizer que esse computador deve manter um **serviço de hospedagem** de sites a qual você é o **cliente** que quer acessá-lo.



Um cliente pode ser qualquer dispositivo que faça acesso a um serviço. E um servidor não precisa, necessariamente, ser um computador robusto, e sim, um simples computador que provê acesso a um serviço, como a hospedagem de páginas Web (um servidor Web). Você pode até mesmo fazer de seu computador um servidor Web (o que você realmente fará quando for testar suas páginas), entretanto não é nada legal ter que deixar seu computador ligado todo o tempo. Por isso precisamos pagar por serviços de hospedagem profissionais.

Para ter acesso a internet você precisa de um endereço, chamado endereço IP (*Internet Protocol*), como por exemplo 192.168.0.1 (isso é um endereço local), 127.0.0.1 (endereço que identifica a máquina local), ou 8.8.8.8 (endereço de um dos servidores do Google). Ter um endereço na internet custa um preço. Talvez você nunca tenha ouvido falar na necessidade de ter que comprar um endereço IP, mas você provavelmente paga as contas da sua operadora de internet. Ela te oferece um endereço IP dinâmico, ou seja, um endereço que muda a cada vez que você se conecta. É claro que a internet não seria viável se precisássemos digitar endereços para cada página, por isso criou-se um serviço que converte endereços de IP em nomes, o chamado DNS (*Domain Name System*). Assim, você não precisa digitar números e sim um **domínio**, como por exemplo: “http://google.com.br”. O *http* indica o protocolo usado na página, mas não se preocupe com ele, o navegador já o insere automaticamente.

Bom, então para se criar um site você precisa:

(1) Primeiro desenvolver sua página (numa das linguagens que em breve lhe apresentaremos);

(2) Conseguir um domínio para seu site (no Brasil, o órgão responsável por isso é o "registro.br");
(3) Conseguir um servidor de hospedagem (um computador que ficará ligado todo o tempo e permitirá que muitas pessoas possam acessar seu site ao mesmo tempo).

Com exceção do item 1, todos os outros itens são pagos. Mas não vamos nos preocupar com isso agora. Podemos simular um serviço de hospedagem instalando um servidor Web na sua máquina (usaremos o Apache). Apache possui um domínio para acesso de seus sites locais, chamado http://localhost. Mas atenção: seu site só ficará acessível na sua máquina.

Quando desenvolvemos um site na nossa máquina chamamos de **ambiente de desenvolvimento**. Ao transferir para um servidor na internet para que outros possam ter acesso chamamos isso de **ambiente de produção**. Neste livro abordaremos apenas o ambiente de desenvolvimento.

Como projetar um Website?

Para projetar um Website deve-se planejá-lo completamente. É importante discutir com uma equipe, ou um orientador(a), os objetivos do seu site. Se ele deve disponibilizar acesso a uma ferramenta, quanto de recurso os usuários externos terão direito de usar ou se seu Website tem o intuito apenas de divulgar resultados.

A projeção e planejamento de um Website deve ocorrer em três fases: (1) *wireframe*, onde são feitos os primeiros desenhos da interface; (2) *mockup*, onde é feito uma representação gráfica final

do Website; e (3) protótipo, onde é feita uma representação funcional da ferramenta.

Wireframe	*Mockup*	Protótipo
Desenho do website. Geralmente feito em um pedaço de papel.	Representação fiel do Website. Geralmente feito em uma ferramenta de design gráfico, como Photoshop.	Versão funcional do Website. Deve apresentar elementos HTML navegáveis, entretanto nem todas as funcionalidades precisam estar disponíveis.

Versões de um site

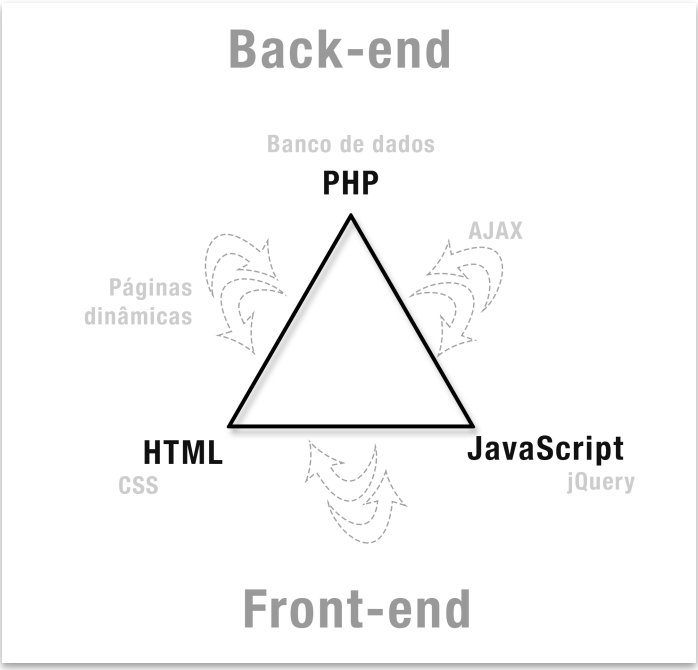
O desenvolvimento de um Website deve passar por algumas versões antes de ser lançado oficialmente.

Alfa	Beta	Final
Primeiras versões do site. Deve ser testada apenas pela a equipe de desenvolvedores do projeto. Versões alfa não devem ser liberadas para o público em geral, pois podem conter falhas gravíssimas que podem comprometer a credibilidade da	Versão para testes. Deve ser testada apenas por um pequeno grupo de usuários. Pode-se realizar testes assistidos, a qual a equipe assiste aos usuários usarem a ferramenta, mas sem opinar; ou testes não assistidos, a qual os usuários testam a ferramenta a distância e	Versão final do site. Deve-se garantir antes da liberação da versão final que *bugs* tenham sido resolvidos. Entretanto, inevitavelmente falhas novas poderão surgir. Falhas graves, geralmente que envolvem a segurança do sistema ou do usuário, devem ser corrigidas

ferramenta desenvolvida.	enviam um *feedback* no final.	imediatamente. Enquanto falhas mais simples podem ser corrigidas em versões ocasionais.

O dogma central do desenvolvimento Web

Assim como a biologia molecular possui um dogma central, proponho aqui um dogma central para melhor explicar o desenvolvimento de sistemas para Web, que denominarei de **Dogma Central do Desenvolvimento Web**.



O desenvolvimento Web é dividido em duas "frontes" de ação: *back-end*, linguagens que rodam diretamente no computador servidor; e *front-end*, linguagens executadas diretamente no computador do usuário.

No dogma central do desenvolvimento Web, as linguagens *back-end* (neste livro será utilizado a linguagem PHP) ficam responsáveis por controlar as requisições do usuário, gerar acesso aos bancos de dados e retornar as páginas HTML devidamente formatadas com folhas em cascata. Essas páginas podem interagir com o ambiente *back-end* por meio das requisições de páginas e dos envios de formulários. As páginas HTML podem ter seu comportamento alterado no *front-end* por *scripts* (aqui representados por arquivos JavaScript e suas bibliotecas, como jQuery). Apesar desses scripts serem carregados pelo servidor, eles podem interagir através de interações assíncronas (como AJAX).

Todas as interações entre *front-end* e *back-end* fazem funcionar o chamado dogma central do desenvolvimento web. Você pode não ter entendido por completo o funcionamento, mas durante o andamento deste livro apresentaremos melhor esses conceitos.

A quem se destina este livro?

O desenvolvimento Web é essencial para divulgação científica. Além disso, o desenvolvimento de ferramentas Web está em ascensão. Assim, este livro se destina a profissionais de diversas formações que aspiram se tornar bioinformatas, como biólogos, geneticistas, biomédicos, microbiologistas, físicos, químicos e até mesmo a profissionais de tecnologia da informação. Esse livro se

destina a todo aquele que quer se aventurar no universo da Bioinformática e deseja ter um guia para seus primeiros passos na programação para Bioinformática utilizando linguagens web.

Assim, tentaremos explicar de maneira simples e didática os princípios básicos de programação através das linguagens PHP e JavaScript. Aprenderemos também a criar páginas HTML e a formatá-las usando folhas de estilo CSS.

Como este livro está organizado?

Na primeira parte apresentaremos a linguagem de marcação HTML e as folhas de estilo em cascata (CSS). Domínio dessas linguagens é importante para construção de páginas da internet. A seguir apresentamos a linguagem de programação *server-side* PHP. Por fim, apresentamos a linguagem *client-side* JavaScript.

Se você leu o primeiro livro da série, *Introdução a programação para Bioinformática com Biopython*, talvez já possa pular para o capítulo 1, mas caso este seja o primeiro livro da série que está lendo, veja a seguir alguns conceitos fundamentais.

Conceitos básicos

Algoritmo: define-se como uma série de passos para que se possa resolver um problema computacionalmente tratável em tempo finito.

Código-fonte: é o código de uma página HTML, suas marcações, folhas de estilo e *scripts*. Diversos navegadores permitem a visualização do código-fonte (geralmente disponível no menu *Exibr* ou clicando com o botão direito e indo até *Inspecionar elemento*). O código PHP não pode ser acessado pelo navegador.

Conceitos básicos

Hardware: é a parte física de um computador: placas, processadores e componentes eletrônicos em geral.

Software: é a parte lógica de um computador, um programa de computador que segue um determinado algoritmo, e que pode ser executável ou interpretável.

Script: conjunto de comandos organizados em um arquivo de texto e que será analisado por um interpretador ou por um compilador.

Função: como o próprio nome já diz é um trecho do código que tem uma função específica, podendo receber parâmetros e retornar informações processadas. Linguagens de programação já possuem funções nativas, o que permite que o programador execute ações sem ter que escrever o algoritmo básico. Entretanto, programadores podem construir suas próprias funções, o que auxilia na organização do código.

Interpretador: constitui num programa de computador que recebe um comando único ou *script* e os converte em linguagem de máquina para que possam sem processados pela CPU (unidade central de processamento, ou simplesmente processador).

Paradigmas: representação de um padrão; metodologia.

Iteração: sinônimo de repetição.

Indentação: estruturação e organização do código. Derivada do inglês *indent*.

Diretório: sinônimo a pasta.

Abstração: na computação pode-se compreender abstração como uma representação simples que explique um processo complexo.

Programação estruturada (PE): método de programação em que o código é estruturado de maneira simples em três partes: sequência, decisão e iteração. A PE pode ser vista como uma programação linear, uma vez que os códigos são executados sequencialmente.

Programação orientada a objetos (POO): na programação orientada a objetos, o código é organizado por classes, que instanciam objetos.

Conceitos básicos

Bloco: define-se bloco como um trecho de código, quase sempre delimitado por chaves "{ }", a qual todo conteúdo está vinculado ao comando inicial.

```
function exemplo_de_bloco( ) {
        # Comandos que pertencem a uma função
}
```

Conseguindo um editor de textos/IDE: Se você leu o livro anterior desta série sabe que para programar precisará de uma IDE ou de pelo menos um editor de textos. Usuários do Linux podem utilizar o editor nativo (*gedit*), que apresenta bons recursos. Recomendamos o *software* **Sublime text** por ser simples, robusto e leve. Se escolher esse editor, sugerimos que experimente o esquema de cores "*monokai*". Esse padrão de cores permite que você programe por horas sem cansar sua visão. Sublime text é compatível com Windows, Linux e MacOS. Sublime text 2 foi utilizado para desenvolvimento dos *scripts* deste livro. **Onde conseguir:** http://www.sublimetext.com

Parte 1

Aparência e conteúdo na Web

Nesta primeira parte apresentamos a linguagem de marcação de textos HTML e as folhas de estilo em cascata (CSS). Dominar os conceitos básicos dessas linguagens é essencial para se tornar um bom programador de páginas da Internet. Demonstraremos ainda como utilizar o *framework* Bootstrap para aperfeiçoar a aparência suas páginas.

1. Introdução ao HTML

HTML (*Hypertext Markup Language* ou na tradução para o português Linguagem de Marcação de Hipertexto) é uma linguagem utilizada para construção de páginas de Internet. HTML define um conjunto de regras que permite a formatação de páginas. Um documento HTML consiste em um arquivo de texto que será lido e interpretado como uma página por *browsers* (navegadores), como por exemplo, Chrome, Firefox, Opera, Edge ou Safari.

Construir páginas em HTML é o principal método para divulgação de conteúdo na Internet. Apesar de haver diversos editores que permitem a construção de páginas sem a necessidade de alterar o código-fonte, compreender a sintaxe dessa linguagem é de grande importância. Dominar a linguagem HTML é essencial para dominar a Web.



Página oficial: https://www.w3.org/html/
Criador: Tim Berners-Lee
Surgiu em: 1993



4. HTML

https://youtu.be/RV7Z2MC8DNI

1.1 História

HTML surgiu no princípio da década de 1990, após Tim Berners-Lee e colaboradores implementarem a comunicação entre máquinas usando o protocolo HTTP. A linguagem HTML foi inspirada na metalinguagem SGML, criada na década 1960 pela IBM. HTML herdou algumas características do SGML para estruturação de texto, como por exemplo o uso de *tags*.

Anos mais tarde, um grupo informal de pesquisadores lançou novas versões do HTML. A partir de 1996, a linguagem HTML passou a ser mantida pela W3C (*World Wide Web Consortium*). A partir da versão 3.5 do HTML, o grupo de pesquisadores passou a trabalhar numa versão do HTML mais simples de ser processada, baseada em XML. Em 1999, surge a versão 4.01, a última especificação lançada pela W3C antes da versão 5.

1.2 HTML5

HTML5 é a mais recente versão da linguagem HTML. Trouxe novas funcionalidades e perspectivas em relação ao papel exercido pelo HTML na Web. HTML traz novas *tags*, como <header> para cabeçalhos, <nav> para menus, <video> e <audio> para

respectivos elementos, além de apresentar uma sintaxe mais simples como por exemplo para declaração do DOCTYPE.

1.3 Sintaxe HTML

Uma página HTML contém marcas, também denominadas *tags,* que fazem a marcação do conteúdo. As *tags* indicam ao navegador como um conjunto de elementos deve ser exibido na tela. *Tags* são delimitadas pelos parênteses angulares: caracteres < (menor que) e > (maior que). Elas ainda podem conter atributos.

As tags podem ser compostas por elementos simples, que não possuem filhos, e elementos compostos, que podem possuir filhos.

Elemento simples:

```
<tag>
```

Elemento composto:

```
<tag>Conteúdo</tag>
```

Além disso, *tags* podem apresentar atributos que agregam novas especificações aos elementos.

```
<tag atributo="valor">Conteúdo</tag>
```

Diz-se que uma *tag* é filha de outra se esta se encontra dentro das delimitações. Observe:

c1_s1.html | https://goo.gl/73s5TX

```
<tag_1>
    <tag_2></tag_2>
    <tag_3></tag_3>
    <tag_4>
</tag_1>
```

No exemplo acima, as *tags* 2, 3 e 4 são filhas da *tag* 1, pois se encontram no mesmo nível. A *tag* 1 é pai das *tags* 2, 3 e 4. Pode ainda dizer que 2, 3 e 4 são *tags* irmãs.

2. Fundamentos do HTML



5. Fundamentos do HTML

https://youtu.be/S-TpzDIGIuY

2.1 Elementos de uma página HTML

Uma página HTML é composta por alguns elementos básicos. São eles: (i) a declaração do tipo do documento; (ii) a *tag* HTML, que armazenará todo conteúdo da página; (iii) o cabeçalho, que armazena informações estruturais da página; e (iv) o corpo, que armazenará os conteúdos e formatações da página.

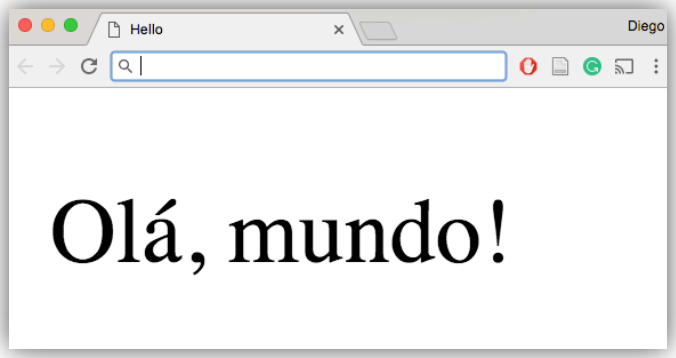
Abaixo vemos um exemplo de uma página salva no formato HTML. Abra seu editor de textos e digite o conteúdo abaixo.

c2_s1.html | https://goo.gl/785N8D

```
<!DOCTYPE html>
<html>
<head>
    <title>Hello</title>
</head>
```

```
<body>
    <p>Olá, mundo!</p>
</body>
</html>
```

Salve o arquivo com o título “s2_c1.html". Página HTML possuem a extensão “.html”. Agora abra o arquivo. O navegador irá exibir a seguinte página:



Nesse exemplo estamos utilizando o navegador Google Chrome. Seu navegador nativo pode ser outro, mas a página exibida deve ser a mesma. Uma das vantagens do desenvolvimento de páginas para internet é a **interoperabilidade**, ou seja, um mesmo código pode ser exibido por diferentes programas da mesma maneira.

Agora vamos entender o código apresentado:

Na primeira linha do *script* é feita a declaração do *doctype* (*document type* ou tipo do documento). A declaração do *doctype* especifica para o navegador qual tipo de documento está sendo lido. Essa informação é fundamental para que os navegadores processem a página corretamente. Para HTML5 é necessário apenas declarar que o tipo de documento é HTML, ou seja, apenas repita essa linha em todos os documentos que forem escritos com HTML5: "<*!doctype html*>".

A seguir é declarada a *tag* <html>. A *tag* <html> especifica o inicio da página escrita na linguagem HTML. Ela deve ser fechada no fim do código. Todo o conteúdo entre a declaração de abertura e fechamento pertence a *tag*. A declaração de *tags* funciona de maneira hierárquica em HTML. Para fechar uma *tag* deve-se utilizar a seguinte sintaxe: </*nome_da_tag*>.

O conteúdo de <html> é dividido por duas outras *tags*: <head> e <body>. Em <head> todas as declarações de informações necessárias para a página são realizadas. Por exemplo, dentro de <head> é feita a declaração da *tag* <title>, que informa que o título da página será "Hello". Dentro da *tag* <body> é inserido o conteúdo que gostaríamos de exibir. Observe que a frase é flanqueada pela *tag* <p>, que indica que o conteúdo se trata de um parágrafo. Observe também como as *tags* são fechadas. A *tag* <p> é filha da *tag* <body>, pois <p> é declarada entre a abertura da *tag* <body> e o fechamento </body>.

2.1.1 Cabeçalho <head>

A *tag* <head> armazena as informações descritivas de um site. As informações armazenadas no cabeçalho não são exibidas pelos navegadores. Além da *tag* de título <title>, o cabeçalho aceita outros tipos de *tags*, como por exemplo:

- **<style>:** marcações de estilo;
- **<link>:** permite a inserção de *links* externos. Usado por exemplo para adição de folhas de estilo CSS;
- **<script>:** permite a inserção de *scripts* na página;
- **<meta>:** permite a inserção de metadados nos documentos, como por exemplo, conjunto de caracteres utilizado (*charset*), nome do autor (*author*), palavras-chave (*keywords*), descrição (*description*) e o tipo de janela para exibição (*viewport*).

No exemplo abaixo demonstraremos como declarar metadados no cabeçalho de uma página HTML.



c2_s2.html | https://goo.gl/HB5zmf

```
<head>
  <meta charset="UTF-8">
  <meta name="description" content="Meu site">
  <meta name="keywords" content="HTML,TAGS">
  <meta name="author" content="Diego Mariano">
  <meta name="viewport" content="width=device-width, initial-
```