Desenvolvimento de Sistemas Web I

Tecnologia em Análise e Desenvolvimento de Sistemas

Paulo Maurício Gonçalves Júnior

Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco

22 de janeiro de 2015

Parte I

Aula Inicial

Roteiro

- Acordo Didático
- Afinal, o que vamos estudar?
- Sobre a disciplina
- Avaliação

Acordo Didático

- Celulares
  - Desligados ou no silencioso
  - Atender apenas fora da sala de aula
- Evitar conversas paralelas
- Sair da sala
  - Evitar sair da sala desnecessariamente
- Atenção
  - Prestar atenção na aula
- Estudo antecipado
  - Não deixar para estudar todo o assunto em cima da hora
- Uso do laboratório
  - Sempre desligar os computadores ao sair
  - Não acessar sites que não têm a ver com a atividade

Desenvolvimento de Sistemas Web I

- Como desenvolver páginas utilizando as técnicas mais modernas.
- Separação de conteúdo, apresentação, e comportamento.
- Desenvolvendo para dispositivos distintos.
- Desenvolvendo para pessoas com necessidades especiais.

Mas afinal, o que vamos estudar?

1. HTML
2. CSS
3. JavaScript
4. Acessibilidade

Critérios de aprovação

Organização Acadêmica Institucional 2014

- **Art. 158** Estará aprovado nos Cursos Superiores, o estudante que obtiver frequência igual ou superior a 75% (setenta e cinco por cento) em cada componente curricular e média 7,0 (sete) em cada componente curricular que componha a matriz do curso.
- **Parágrafo Único** O estudante dos Cursos Superiores que tiver menos de 75% (setenta e cinco por cento) de frequência em cada componente curricular, independente da média alcançada, estará reprovado, sem direito ao exame final.

Segunda chamada

- A prova de segunda chamada será aplicada ao final do semestre letivo.
- Ela serve para os alunos que faltaram a uma das provas de unidade por motivo justificado.
- Ela substitui somente **uma** das provas.

Atenção!

- Qualquer aluno **pode** ser chamado a resolver as questões das provas, a critério do professor. Caso não consiga responder às questões, sua nota será cancelada.
- O único método aceito para resolução das provas, trabalhos e listas de exercícios é o conhecimento, que é o que será avaliado. Cópias, sorte, psicografia não serão aceitos como métodos válidos, levando à anulação da avaliação.
- As provas são uma forma de avaliação (quantitativo), mas não a única. Participação em sala de aula, tirar dúvidas por e-mail ou presencialmente nos encontros de atendimento ao aluno, frequência, pontualidade, também contam (qualitativos). Alunos que possuem critérios qualitativos baixos e quantitativos altos são preferencialmente chamados a resolver suas provas.

Dúvidas, perguntas, sugestões...

- `http://sites.google.com/site/paulomgj/home/dsw1`.
- A página da disciplina possui informações tais como:
  - conteúdo apresentado em cada aula;
  - exercícios propostos e resolvidos;
  - horário das aulas;
  - e-mail do professor para contato;
  - plano de ensino.

Parte II

O Modelo de Padrões Web – HTML, CSS e JavaScript

Marcação, a base de toda página

- HTML é uma linguagem de marcação composta de elementos, que contém atributos.
- Esses elementos são usados para marcar os diferentes tipos de conteúdo existente nos documentos (seções, parágrafos, listas, etc.).
- Atributos definem informações extras sobre os elementos.
- A marcação deve ser o mais semântica possível, ou seja, descrever a função do conteúdo o mais precisa possível.

```
<div id="footer"></div>
```

CSS I

- CSS permite que você controle a apresentação e o layout da página.
- Pode-se mudar ou adicionar as cores, cores de fundo, tamanho de fontes, posicionamento, entre outras.
- Existem três formas de aplicar CSS:
  1. Redefinir um elemento:

```
p {
  line-height: 2;
  color: green;
}
```

  2. Definir um identificador

```
<p id="highlight"></p>
```

```
#highlight {
  line-height: 2;
  color: green;
}
```

CSS II

3. Definir uma classe

```
<p class="highlight"></p>
<p class="highlight"></p>
```

```
.highlight {
  line-height: 2;
  color: green;
}
```

JavaScript

- Permite que você adicione comportamento às suas páginas. Pode ser usado para validar dados de formulários, arrastar e soltar elementos, modificar o estilo dinamicamente, etc.

Por que separados? I

- É possível obter conteúdo, estilo e layout somente usando HTML. Por que então usar HTML e CSS?
- Razões principais:
  1. **Eficiência do código:** Quanto maior o código, mais tempo demorará para o arquivo ser transferido. O indicado é criar um código HTML compactado e incluir o estilo e apresentação em arquivos CSS separados.
  2. **Facilidade de manutenção:** Se seu estilo e apresentação estão especificados em um único local, fica mais fácil realizar modificações.
  3. **Acessibilidade:** Documentos criados de forma semântica, ao invés de focado na apresentação, são mais facilmente navegáveis e a informação neles é mais facilmente encontrada pelo usuário.
  4. **Compatibilidade entre dispositivos:** Sendo a página HTML somente conteúdo semanticamente marcado, sem informação de estilo, ele pode ser reformatado para dispositivos distintos aplicando-se estilo alternativos.

Por que separados? II

5. **Engenhos de busca:** Engenhos de busca usam programas que leem páginas e as indexam. Se estes programas tem dificuldade em encontrar o conteúdo de sua página, ou não interpreta corretamente o que é importante, provavelmente seu ranking na página será impactado negativamente.

- Estudar sobre a história da internet, da web e a evolução dos padrões web [2]; como a internet funciona [1]; e ler mais sobre esta aula [3].

Referências

[1] Web Education Community Group. How does the Internet work, 2011. URL http://www.w3.org/community/webed/wiki/How_does_the_Internet_work.

[2] Web Education Community Group. The history of the Web, 2012. URL http://www.w3.org/community/webed/wiki/The_history_of_the_Web.

[3] Web Education Community Group. The web standards model - HTML CSS and JavaScript, 2012. URL http://www.w3.org/community/webed/wiki/The_web_standards_model_-_HTML_CSS_and_JavaScript.

Parte III

Utilização de Padrões Web Atualmente

Introdução

- Padrões web permitem a interoperabilidade entre diferentes navegadores, em diferentes sistemas operacionais, em qualquer dispositivo eletrônico disponível.
- Como é a realidade? Os navegadores são 100% compatíveis com os padrões? Os desenvolvedores estão usando os padrões corretamente?
- A resposta é **não**. Isto é o ideal, mas ainda longe da realidade.

Como se verifica a compatibilidade com os padrões?

- Páginas que seguem os padrões não parecem diferente de páginas que não seguem os padrões.
- O código da página é mais limpo e organizado, sem formatação incluída.
- A forma mais fácil de identificar se uma página segue os padrões é usar um validador.
- Como verificar se os navegadores são compatíveis?
- Alguns testes verificam o suporte dos navegadores para alguns padrões [1, 2].

Compatibilidade dos sites atualmente

- Vejamos como sites populares se saem nos testes de validação.
- Quanto maior o site, mais complicado é deixá-lo válido.

Por que tão poucos sites são compatíveis?

- Educação
  - Cursos não ensinam como desenvolver sites compatíveis.
- Razões de negócio
  - Navegadores apresentam código sem estar validado. Para que gastar tempo e dinheiro validando?

Referências

[1] Niels Leenheer. HTML5test, 2015. URL https://html5test.com/.

[2] The Web Standards Project. Acid Tests, 2015. URL http://www.acidtests.org/.

Parte IV

Introdução a HTML

O que é HTML

- HTML é uma linguagem para descrever o conteúdo de páginas web. Ela usa uma sintaxe especial contendo marcadores (chamados "elementos") que englobam porções de texto para indicar como agentes do usuário devem interpretar aquela porção do documento.

HTML parece com o que

- HTML é apenas uma representação textual de conteúdo e seu significado geral.
- Exemplo de um elemento:

```
<h2 id="htmllooks">What HTML looks like</h2>
```

- `<h2>` é um marcador (tag) que significa "o que segue deve ser considerado uma seção de nível 2".
- `</h2>` é a tag de fechamento.
- A tag de abertura, de fechamento, e tudo dentro delas é chamado "elemento".
- `id="htmllooks"` é um atributo.

A história de HTML

- Quando Tim Berners-Lee inventou a World Wide Web, ele criou o primeiro servidor web, navegador e a primeira versão de HTML.
- Muitos dos elementos da primeira versão ainda existem.
- Com mais pessoas escrevendo páginas e navegadores, mais características foram sendo acrescentadas a HTML. Isso levou à necessidade de padronização.

Versão HTML	**Responsável**	**Ano**
1.0	IETF	1993–94
2.0	Dave Raggett	1995
3.0	W3C	1996
4.0	W3C	1997
4.01	W3C	1999
XHTML 1	W3C	2000
HTML 5	W3C	2014

A estrutura de um documento HTML

- O menor documento válido parece com o que segue:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Example page</title>
  </head>
  <body>
    <h1>Hello world</h1>
  </body>
</html>
```

A sintaxe dos elementos HTML

- Um elemento básico em HTML consiste em dois marcadores englobando uma porção de texto. Existem elementos que não englobam texto, e praticamente todos podem conter sub-elementos.
- Todos os atributos seguem a forma `nome="valor"`

```
<div id="masthead">
  <h1>The Basics of
    <abbr title="Hypertext Markup Language">HTML</abbr>
  </h1>
</div>
```

A sintaxe dos elementos HTML

- <abbr> é considerado um filho de <h1>, que por sua vez é filho de .
- Por sua vez, é pai de <h1>, que é pai de <abbr>.
- Este conceito de herança é importante, pois forma a base de CSS e é bastante utilizado por JavaScript.

Elementos de bloco e de linha

- Existem duas categorias gerais de elementos em HTML, que correspondem aos tipos de conteúdo e estrutura que estes elementos representam – elementos de nível de bloco (*block*) e elementos de nível de linha (*inline*).
- Elementos de nível de bloco representam elementos de nível mais alto, normalmente informando a estrutura do documento. Elementos de bloco comuns são parágrafos, itens de lista, seções, e tabelas.
- Elementos de linha são aqueles que estão contidos em elementos de bloco e englobam apenas pequenas partes do conteúdo do documento. Elementos de linha comuns incluem links, palavras ou frases destacadas e pequenas citações.

Caracteres especiais

- Em HTML, os caracteres $<$, $>$ e & são especiais. Os dois primeiros servem para começar e finalizar tags.
- & serve para incluir caracteres especiais. O ampersand (&) introduz a referência e o ponto-e-vírgula finaliza.

O que é XHTML

- Serve para escrever um código HTML utilizando as regras de formação de XML: XHTML = HTML + XML [1, 3].

HTML	**XHTML**
Elementos e atributos são insensíveis à caixa, <h1> é o mesmo que <H1>	Elementos e atributos são sensíveis; todos devem estar em minúsculas
Certos elementos não precisam da tag de fechamento (<p>), enquanto outras não tem <img>	Todos os elementos devem ser fechados explicitamente.
Alguns atributos não precisam estar entre aspas	Todos os atributos devem estar entre aspas
Resumos podem ser usados para alguns atributos <option selected>	Todos os atributos devem seguir a forma completa <option selected ="selected">
Servidores devem enviar os documentos com o tipo text/html	Deve enviar um dos seguintes tipos application/xhtml+xml, text/xml ou application/xml

- Mais sobre esta aula em [2]

Referências

[1] W3C. HTML5 - A vocabulary and associated APIs for HTML and XHTML, 2014. URL http://www.w3.org/TR/html5/.

[2] Web Education Community Group. The basics of HTML, 2014. URL http://www.w3.org/community/webed/wiki/The_basics_of_HTML.

[3] WHATWG. HTML vs. XHTML, 2011. URL https://wiki.whatwg.org/wiki/HTML_vs._XHTML.

Parte V

Validando seu HTML

Introdução I

- Após a criação de algumas páginas HTML, como saber se elas estão corretas e sem erros, de forma a garantir que essa página seja visualizada de forma correta pelos diversos navegadores?
- Validação é a resposta! Programa utilizado para verificar se os documentos estão seguindo as regras descritas nas recomendações do W3C.
- Identifica erros como falta de fechamento de tags, falta de aspas, dentre outros.
- O W3C oferece alguns validadores:
  - *The W3C MarkUp Validation Service* [2]: analisa o documento (X)HTML e verifica se o mesmo segue a recomendação.

Introdução II

- *The W3C Link Checker* [3]: testa todos os links presentes na página e verifica se eles apontam para recursos ativos.
- *The W3C CSS Validation Service* [1]: verifica se os documentos CSS estão de acordo com a recomendação.

- Veremos o primeiro deles hoje.

Erros

- Em programação, existem de forma geral dois tipos de problemas com o código:
  - Erros de sintaxe: estes acontecem quando um erro na escrita do código torna o computador incapaz de executar ou compilar o programa corretamente.
  - Erros de lógica: estes acontecem quando o código não reflete o intuito do programador.
- O primeiro tipo de erro é o mais fácil de ser corrigido.
- Mas HTML **não** é uma linguagem de programação. Navegadores apresentam páginas mesmo quando elas possuem erros de sintaxe, tentando interpretar a vontade do desenvolvedor. Este é um dos motivos do rápido crescimento da WWW.

Por que validar?

- Duas razões fortes:
  - Você não é sempre perfeito, nem seu código – todos cometemos erros, e sua página terá maior qualidade (ou seja, trabalhar de forma mais consistente) se você remover os erros.
  - Navegadores mudam. No futuro, é provável que os navegadores sejam mais estritos na interpretação de código inválido, e não o contrário.
- Como diz o ditado: *Aprenda as regras de forma a saber quebrá-las apropriadamente.*

Navegadores diferentes interpretam HTML inválido de forma diferente

- HTML válido é seu único contrato com os fabricantes de navegadores. A especificação HTML diz como você deve escrever, e como eles devem interpretar seu documento.
- Se o navegador receber código inválido, ele tentará corrigi-lo. Mas diferentes navegadores interpretam erros de forma distinta.
- Como interpretar o código abaixo?

```
<p><strong>This text should be bold</p>
<p>Should this text be bold? How does the HTML look when rendered?</p>

<p><a href="#"></strong>This text should be a link</p>
<p>Should this text be a link? How does the HTML look when rendered?</p
    >
```

Como validar suas páginas?

- Ir para a página do W3C HTML Validator.
- Informar a URL da página a ser validada, enviar uma página local para validação, ou informar a página diretamente.
- Interpretar o primeiro erro, corrigi-lo e revalidar. Muitas vezes um erro gera várias indicações de erro no validador. Corrigir um por um é mais indicado.
- Mais sobre esta aula em [4]

Referências

[1] W3C. CSS Validation Service, 2015. URL http://jigsaw.w3.org/css-validator/.

[2] W3C. The W3C MarkUp Validation Service, 2015. URL http://validator.w3.org/.

[3] W3C. The W3C Link Checker, 2015. URL http://validator.w3.org/checklink/.

[4] Web Education Community Group. Validating your HTML, 2012. URL http://www.w3.org/community/webed/wiki/Validating_your_HTML.

Parte VI

Doctype – Tipo do Documento

Escolhendo o Doctype correto

- A primeira linha de cada documento HTML deve conter a DTD (*Document Type Definition*).
- Ele define que elementos e atributos podem ser usados por uma determinada versão de HTML.
- Ele é usado pelos navegadores para saber que modo de renderização utilizar e pelos validadores para checar seu documento.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR
    /html4/strict.dtd">
```

Doctype switching e modos de renderização

- A maioria dos navegadores utiliza o doctype para saber como renderizar a página.
- A partir desta informação eles tem noção se os desenvolvedores tiveram cuidado ou não na criação das páginas.
- Caso positivo, eles usam um modo chamado “modo padrão”. Caso não, eles usam o “modo quirks” (compatibilidade).

Escolhendo um doctype

- Existem três versões para ambos HTML 4.01 e XHTML 1.0: strict, transitional, e frameset.
- (X)HTML strict é caracterizado pela sua proibição das chamadas tags depreciadas.
- A única diferença entre transitional e frameset, já que ambos permitem tags depreciadas, é que o último permite frames.
- Você informa qual versão e sabor está usando em seu documento pela utilização de uma declaração DOCTYPE. Uma vez que a informação faça parte de sua página, você pode usar um validador para determinar se o código usado em sua página corresponde ao código permitido para aquela versão e sabor. Validadores são uma ótima forma para checar por erros e em geral, garantir que seu código está correto.

DOCTYPEs

- XHTML 1.0

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "
    http://www.w3.org/TR/xhtml/DTD/xhtml1-transitional.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://
    www.w3.org/TR/xhtml/DTD/xhtml1-strict.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http
    ://www.w3.org/TR/xhtml/DTD/xhtml1-frameset.dtd">
```

- HTML 4.01

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
    http://www.w3.org/TR/html4/loose.dtd">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.
    org/TR/html4/strict.dtd">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http
    ://www.w3.org/TR/html4/frameset.dtd">
```

- HTML 5

```
<!DOCTYPE html>
```

HTML 5

- É compatível com as versões já existentes.
- Adiciona características poderosas a HTML que antes só estavam disponíveis através de plugins, ou de códigos Javascript complexos.
- Mais adequado a aplicações dinâmicas do que HTML 4.
- Possui um algoritmo de parsing claramente definido de forma que todos os navegadores criem um mesmo DOM para a mesma marcação, independente de validação.

A declaração XML

- Caso se esteja utilizando XHTML, é possível existir uma declaração XML antes da DTD.

```
<?xml version="1.0" encoding="UTF-8"?>
```

- Só é obrigatório utilizá-la caso se esteja usando XHTML e o servidor enviando como XML e a codificação dos caracteres é diferente de UTF-8 e o servidor não está determinando a codificação dos caracteres.
- Mais sobre esta aula em [1]

Referências

[1] Web Education Community Group. Doctypes and markup styles, 2012. URL http://www.w3.org/community/webed/wiki/Doctypes_and_markup_styles.

Parte VII

Elemento `head`

Elemento head

- Praticamente nada que você coloca no elemento <head> fica visível para o usuário.
- Este elemento é o local onde a maioria das instruções para o navegador está presente e onde você armazena informações extras – chamadas meta-informações – sobre o documento.

Julgando um documento pelo seu título

- O texto dentro do elemento <title> geralmente é mostrado na barra de títulos.
- É a primeira informação que os usuários veem.
- Escreva um título conciso sobre o que o documento é.

```
<!DOCTYPE html>
<html>
<head>
  <title>I am a title example</title>
</head>
<body>
</body>
</html>
```

Adicionando palavras-chave e descrição

- Usado por engenhos de busca na indexação do documento e pelo navegador para adicionar informação aos favoritos.

```
<head>
  <title>Yahoo! UK & Ireland Eurosport - Sports News | Live Scores |
      Sport</title>
  <meta name="description" content="Latest sports news and live scores
      from Yahoo! Eurosport UK. Complete sport coverage with Football
      results, Cricket scores, F1, Golf, Rugby, Tennis and more.">
  <meta name="keywords" content="eurosport,sports,sport,sports news,
      live scores,football,cricket,f1,golf,rugby,tennis,uk,yahoo">
</head>
```

Adicionando estilos

- Pode-se colocar estilos que serão aplicados ao documento.
- Pode-se usar o atributo `media` para indicar o meio que usará os estilos.
  - `screen`: Monitores
  - `print`: Impressão
  - `handheld`: Dispositivos móveis
  - `projection`: Apresentações em HTML

```
<style type="text/css">
  body{
    background:#000;
    color:#ccc;
    font-family: helvetica, arial, sans-serif;
  }
</style>
```

Adicionando conteúdo dinâmico com JavaScript

- No cabeçalho também podem se encontrar os scripts a serem executados pela página.
- Quando o navegador encontra um script, ele para as outras atividades e tenta interpretar o script.

```
<head>
<script>
  function leave(){
    return confirm("This will take you to another site,\n are you sure
        you want to go?")
  }
</script>
</head>
<body>
Test!
<a href="http://dailypuppy.com" onclick="return leave()">The Daily
    Puppy</a>
</body>
```

Externalizando

- O ideal é não colocar estilos e scripts dentro das páginas.
- Facilita a manutenção e deixa as páginas mais simples e rápidas de baixar.

```
<head>
  <title>Breeding Dogs - Tips about Alsatians</title>
  <meta name="description" content="How to breed Alsatians, tips on
      proper breeding and information about common issues with this
      breed.">
  <meta name="keywords" content="Dogs,Alsatian,Breeding,Dog,Tips,Free,
      Pet">
  <link rel="stylesheet" type="text/css" media="screen" href="styles.
      css">
  <link rel="stylesheet" type="text/css" media="print" href="
      printstyles.css">
  <script src="leaving.js"></script>
</head>
```

O elemento `link`

- Define a ligação entre o documento e um recurso externo.
- Atributo `rel`: especifica a relação entre o documento atual e o recurso externo.
- Exemplos:
  - `icon`: Importa um ícone para representar o documento [1].
  - `stylesheet`: Importa uma folha de estilo a ser aplicada ao documento.
  - `prev`: Indica que o documento faz parte de uma série, e que o documento prévio na série é o documento referenciado.
  - `next`: Indica que o documento faz parte de uma série, e que o próximo documento na série é o documento referenciado.

```
<link rel="icon" type="image/x-icon" href="icone.ico">
<link rel="stylesheet" type="text/css" href="styles.css">
<link rel="prev" href="http://www.example.com/arti-part1.html">
<link rel="next" href="http://www.example.com/arti-part3.html">
```

- Mais sobre esta aula em [2, 3]

Referências

[1] F. Generator.com. Favicon Generator, 2015. URL http://www.favicongenerator.com/.

[2] Web Education Community Group. The HTML head element, 2014. URL http://www.w3.org/community/webed/wiki/The_HTML_head_element.

[3] Web Education Community Group. More about the document head, 2014. URL http://www.w3.org/community/webed/wiki/More_about_the_document_head.

Parte VIII

Formatação HTML Básica

Espaço em branco

- Em HTML múltiplos espaços são interpretados como um único espaço.

```
<h3>In   the
                beginning</h3>
<h3>In the beginning</h3>
```

Elementos de nível de bloco

Seções de página

- Uma vez que a página foi quebrada em seções lógicas, cada seção deve ser introduzida por um cabeçalho específico.
- HTML define seis níveis de cabeçalhos `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, e `<h6>` (do de maior importância para o menor).

```
<h1>Marking up textual content in HTML</h1>
<h2>Introduction</h2>
<h2>Space -- the final frontier</h2>
<h2>Block level elements</h2>
<h3>Page section headings</h3>
<h3>Generic paragraphs</h3>
<h3>Quoting other sources</h3>
<h3>Preformatted text</h3>
<h2>Inline elements</h2>
```

Elementos de nível de bloco

Parágrafos

- O parágrafo é o bloco básico da maioria dos documentos.

```
<p>This is a very short paragraph. It only has two sentences.</p>
```

Elementos de nível de bloco
Referenciando texto

- Usado para referenciar um texto externo.
- Referências de nível de bloco usar `<blockquote>`. Pode-se usar o atributo `cite` para indicar o endereço de onde se encontra a referência.
- Referências de nível de parágrafo usar `<q>`.

```
<blockquote>
<p>Para ser grande, sê inteiro: nada<br />
Teu exagera ou exclui.<br />
Sê todo em cada coisa. Põe quanto és<br />
No mínimo que fazes.<br />
Assim em cada lago a lua toda<br />
Brilha, porque alta vive.</p>
</blockquote>
<p>Poema de Fernando Pessoa.</p>
```

Elementos de nível de bloco

Usando texto pré-formatado

- Usado para representar códigos fonte que ocupam várias linhas através do comando <pre>.

```
<p>O código abaixo apresenta um exemplo básico da linguagem C:</p>
<pre>
#include <stdio.h>

int main(int argc, char** argv) {
  printf("Hello World!");
  return 0;
}
</pre>
```

Elementos de nível de bloco

Estrutura da página

- `<header>`: Usado para representar o cabeçalho da página ou de um conteúdo.
- `<footer>`: Usado para representar o rodapé da página ou de um conteúdo.
- `<nav>`: Contém o menu de navegação, ou outra funcionalidade de navegação da página.
- `<article>`: Contém o conteúdo da página.
- `<section>`: Usado tanto para agrupar diferentes áreas de funcionalidades ou assuntos ou para definir as diferentes seções de um conteúdo.
- `<aside>`: Define um bloco de conteúdo que está relacionado com o conteúdo principal, mas não é central ao fluxo.

Elementos de linha
Referências curtas

- São usadas em uma sentença normal ou parágrafo e usam o elemento `<q>`. Também pode conter o atributo `cite`.
- Geralmente fica entre aspas duplas.

```
<p>This did not end well for me. Oh well, <q lang="fr">c'est la vie</
    q> as the French say.</p>
```

Elementos de linha
Indicando ênfase

- Para indicar uma ênfase forte usar `<strong>`.
- Para indicar uma ênfase fraca usar `<i>` ou `<em>`.

```
<p><em>Please note: the kettle <strong>must</strong> be unplugged every
     evening, otherwise it will explode - <strong>killing us all</
    strong></em>.</p>
```

Elementos de linha

Elementos de apresentação – não usar

- font, b, strike, sup, u, tt, big, small.

Elementos de linha
Usando fonte monospace

- Fonte monospace é aquela em que todos os caracteres possuem o mesmo tamanho (como em uma máquina de escrever).
- Para representar códigos fonte usar `<code>`.
- Para texto monospace em geral usar `<tt>`.
- Para instruções do teclado usar `<kbd>`.
- Para amostra de texto usar `<samp>`.

```
<p><code>#!usr/local/bin/perl</code></p>
```

- Estudar mais em [1–3].

Referências

[1] Web Education Community Group. Lesser - known semantic elements, 2012. URL http://www.w3.org/community/webed/wiki/Lesser_-_known_semantic_elements.

[2] Web Education Community Group. HTML structural elements, 2013. URL http://www.w3.org/community/webed/wiki/HTML_structural_elements.

[3] Web Education Community Group. Marking up textual content in HTML, 2014. URL http://www.w3.org/community/webed/wiki/Marking_up_textual_content_in_HTML.

Parte IX

Listas em HTML

Listas

- Listas são usadas para agrupar pedaços de informação, de forma que eles fiquem claramente associados uns com os outros e facilite a leitura.
- Listas ajudam a criar documentos mais bem estruturados, mais acessíveis, e mais fáceis de manter.
- Existem três tipos de listas:
  - Listas não ordenadas
  - Listas ordenadas
  - Listas de definição

Listas não ordenadas

- Usadas para agrupar um conjunto de itens relacionados, que podem aparecer em qualquer ordem.

```
<ul>
  <li>bread</li>
  <li>coffee beans</li>
  <li>milk</li>
  <li>butter</li>
</ul>
```

Listas ordenadas

- Usadas para agrupar um conjunto de itens relacionados, que devem aparecer em uma ordem específica. Um exemplo é uma receita.
- Podem aparecer com diversos tipos de numeração – números ou letras.

```
<ol>
  <li>Gather ingredients</li>
  <li>Mix ingredients together</li>
  <li>Place ingredients in a baking dish</li>
  <li>Bake in oven for an hour</li>
  <li>Remove from oven</li>
  <li>Allow to stand for ten minutes</li>
  <li>Serve</li>
</ol>
```

Listas ordenadas

- Podemos começar uma lista ordenada a partir de uma determinada posição, através do atributo `start`.

```
<ol start="4">
  <li>Bake in oven for an hour</li>
  <li>Remove from oven</li>
  <li>Allow to stand for ten minutes</li>
  <li>Serve</li>
</ol>
```

Listas de definição

- Associa itens específicos à sua definição.

```
<dl>
  <dt>Term</dt>
  <dd>Definition of the term</dd>
  <dt>Term</dt>
  <dd>Definition of the term</dd>
  <dt>Term</dt>
  <dd>Definition of the term</dd>
</dl>
```

- <dl> define a lista, <dt> define o termo, e <dd> define a descrição.

A diferença entre listas e textos

- Se você precisa modificar a ordem em uma lista ordenada, basta mover o elemento dentro da lista. Se foi escrito textualmente, a ordem deve ser alterada manualmente.
- Permite modificar o estilo através de CSS.
- Utiliza a estrutura semântica correta, facilitando a acessibilidade.

Listas aninhadas

- Uma lista aninhada deve estar associada a um item. Dentro do código, a lista aninhada está dentro de um item.

```
<ol>
  <li>Chapter One
    <ol>
      <li>Section One</li>
      <li>Section Two </li>
      <li>Section Three </li>
    </ol>
  </li>
  <li>Chapter Two</li>
  <li>Chapter Three  </li>
</ol>
```

- Estudar mais em [1, 2].

Referências

[1] Web Education Community Group. HTML lists, 2012. URL http://www.w3.org/community/webed/wiki/HTML_lists.

[2] Web Education Community Group. Creating multiple pages with navigation menus, 2012. URL http://www.w3.org/community/webed/wiki/Creating_multiple_pages_with_navigation_menus.

Parte X

Imagens em HTML

Imagens

- São ótimas formas de passar informações complexas.
- Nem todos podem ver imagens
  - Pessoas podem desabilitar imagens para agilizar a navegação (principalmente em dispositivos móveis)
  - Usuários podem ser cegos ou possuir problemas de visão.
  - Visitantes podem ser de culturas diferentes e não entender as imagens.
  - Engenhos de busca analisam texto e descartam figuras.

Tipos de Imagens

- Se a imagem é crucial para o conteúdo da página, ele deve ser incluído através do elemento <img> com um texto alternativo apropriado.
- Se a imagem for usada para embelezamento, deve-se usar imagens de plano de fundo com CSS. Assim não precisamos de conteúdo alternativo e temos mais opções de lidar com a figura em CSS.

O elemento img e seus atributos

- O elemento `<img>` deve ser utilizado para incluir figuras na página.
- O atributo `src` informa o endereço da figura.
- O atributo `alt` informa o texto alternativo a ser utilizado caso a figura não seja carregada.

```
<img src="balconyview.jpg" alt="View from my balcony, showing a row of
    houses, trees and a castle">
```

O elemento img e seus atributos

- O atributo `title` contém um texto que será apresentado quando o mouse passar por cima da figura.
- O atributo `longdesc` contém o endereço de uma página com uma descrição mais detalhada da figura.

```
<img src="balconyview.jpg" alt="View from my balcony, showing a row of
    houses, trees and a castle" title="Shown when the mouse passes over
     the image" longdesc="balcony-longdesc.html">
```

O elemento img e seus atributos

- Usando os atributos `width` e `height`, o navegador já aloca a posição e o espaço necessário na página, evitando que a página mude de aparência.
- Cada navegador redimensiona de uma maneira diferente.
- Imagens são elementos de linha.
- Figuras de fundo são aplicadas usando CSS.
- Estudar mais em [1].

Referências

[1] Web Education Community Group. Images in HTML, 2013. URL http://www.w3.org/community/webed/wiki/Images_in_HTML.

Parte XI

Links

Links

- Links são partes de uma página que apontam para outros recursos – outros documentos HTML, textos, arquivos PDF, etc.
- Alguns são seguidos automaticamente pelo navegador (elemento `link`) e outros são opcionais, só sendo ativados quando o usuário clica neles. Estes são chamados de âncoras e podemos adicioná-lo usando o elemento `<a>`.

Links

- Uma âncora possui vários atributos:
  - `href` – o recurso para o qual ele aponta.
  - `id` – identificador da âncora se ela for um alvo e não um link.
  - `title` – informações extras sobre o link.

```
<p><a href="http://developer.yahoo.com">Yahoo Developer Network</a></p>
```

Link ou alvo?

- O elemento `<a>` pode desempenhar diversos papéis dependendo dos atributos que forem utilizados. O mais comum é o atributo `href`, que define para qual recurso ele aponta. Este atributo pode conter os seguintes valores:
  - Uma URL na mesma pasta, relativa a pasta atual, ou absoluto em relação à raiz do servidor.
  - Uma URL em um servidor distinto.
  - Um identificador de fragmento ou nome `id` precedido por uma barra.
  - Uma mistura de URL e identificador de fragmento – você pode ligar diretamente a uma seção de um documento distinto.

Informações extras

- Podemos usar o atributo `title` para prover informações extras, mas não informações cruciais, pois leitores de tela não leem esta informação de forma padrão, como:
  - Ligações a recursos não-HTML como arquivos PDF, imagens, vídeos, sons ou downloads: permite ao usuário acessar ou não o recurso, baseado na capacidade de interpretação do conteúdo, no tamanho do conteúdo, etc.
  - Ligações para páginas em um servidor distinto.
  - Ligação para um documento que abrirá em um frame diferente ou pop-up.

Frames

- Frames: permite carregar partes de uma página, enquanto outras partes permanecem disponíveis.
- Problemas:
  - Engenhos de busca não indexam completamente a página, podendo indexar suas partes que, quando apresentadas, perdem o contexto.
  - Não se consegue armazenar as páginas nos favoritos: sempre começará na página principal.
  - Usuários que dependem de tecnologia assistiva tem dificuldade em navegar em páginas com frames.
- Links em um frame usam o atributo `target` para indicar em qual frame o conteúdo do link deve ser aberto. Caso não exista o frame específico, uma nova página é aberta.
- Em resumo: **nunca usar frames**.

Benefícios de links

- Benefícios de ligações para sites de terceiros:
  - Credibilidade: não tem receio de perder os clientes.
  - Apresentar produtos ou serviços que seu site não contempla.
  - Referenciar um artigo ou produto, apresentando sua solução diferenciada.
- Os textos dos links são importantes: palavras distintas devem levar a páginas distintas (evitar textos como *clique aqui*).
- Estudar mais em [1].

Referências

[1] Web Education Community Group. HTML links - lets build a web, 2012. URL http://www.w3.org/community/webed/wiki/HTML_links_-_lets_build_a_web.

Parte XII

Tabelas em HTML

Tabelas em HTML

- Usadas para organizar dados em formato tabular.
- Antigamente eram usadas para posicionar elementos na tela: geravam arquivos maiores, mais difíceis de manter, e difíceis de modificar.

Exemplo

```
<table>
  <tr>
    <td>Volcano Name</td>
    <td>Location</td>
    <td>Last Major Eruption</td>
    <td>Type of Eruption</td>
  </tr>
  <tr>
    <td>Mt. Lassen</td>
    <td>California</td>
    <td>1914-17</td>
    <td>Explosive Eruption</td>
  </tr>
  <tr>
    <td>Mt. Hood</td>
    <td>Oregon</td>
    <td>1790s</td>
    <td>Pyroclastic flows and Mudflows</td>
  </tr>
</table>
```

Tabelas em HTML

- <table>: indica ao navegador que o conteúdo será organizado no formato tabular.
- <tr>: indica uma linha na tabela.
- <td>: indica uma célula na tabela.

Mais funcionalidades

```
<table>
  <caption>Recent Major Volcanic Eruptions in the Pacific Northwest</
      caption>
  <tr>
    <th>Volcano Name</th>
    <th>Location</th>
    <th>Last Major Eruption</th>
    <th>Type of Eruption</th>
  </tr>
  <tr>
    <td>Mt. Lassen</td>
    <td>California</td>
    <td>1914-17</td>
    <td>Explosive Eruption</td>
  </tr>
  <tr>
    <td>Mt. Hood</td>
    <td>Oregon</td>
    <td>1790s</td>
    <td>Pyroclastic flows and Mudflows</td>
  </tr>
</table>
```

Mais funcionalidades

- <caption>: permite informar o título da tabela. Os navegadores tendem a centralizar o título e usar a largura da tabela. Pode aparecer acima ou abaixo da tabela (através de CSS).
  - Dá um título descritivo resumido da tabela de dados.
  - É apresentado na visão do navegador.
  - É facilmente identificado por tecnologias assistivas.
  - É encontrado por engenhos de busca.
- <th>: indica o título de cada coluna da tabela. Os navegadores tendem a tornar o conteúdo em negrito.

E ainda mais funcionalidades I

```
<table summary="a summary of recent major volcanic eruptions in the
    Pacific Northwest">
<caption>Recent Major Volcanic Eruptions in the Pacific Northwest</
    caption>
  <thead>
    <tr>
      <th scope="col">Volcano Name</th>
      <th scope="col">Location</th>
      <th scope="col">Last Major Eruption</th>
      <th scope="col">Type of Eruption</th>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <td colspan="4">Compiled in 2008 by Ms Jen</td>
    </tr>
  </tfoot>
  <tbody>
    <tr>
```

E ainda mais funcionalidades II

```
      <th scope="row">Mt. Lassen</th>
      <td>California</td>
      <td>1914-17</td>
      <td>Explosive Eruption</td>
    </tr>
    <tr>
      <th scope="row">Mt. Hood</th>
      <td>Oregon</td>
      <td>1790s</td>
      <td>Pyroclastic flows and Mudflows</td>
    </tr>
    <tr>
      <th scope="row">Mt. St. Helens</th>
      <td>Washington</td>
      <td>1980</td>
      <td>Explosive Eruption</td>
    </tr>
  </tbody>
</table>
```

E ainda mais funcionalidades I

- Elemento `<thead>` – define a seção de cabeçalho da tabela de dados. Sua tag de abertura é colocada diretamente após a tag de fechamento `<caption>` e diretamente antes da primeira tag de abertura da linha `<tr>`.
- Elemento `<tfoot>` – define a seção de rodapé da tabela de dados. É opcional. Se usá-lo, deve aparecer diretamente antes da tag de abertura do corpo da tabela `<tbody>`.
- Elemento `<tbody>` – define o corpo da tabela e engloba seu conteúdo. Vem diretamente após o fechamento do elemento `</tfoot>`, e antes do elemento de abertura de linha `<tr>`.
- `colspan, rowspan`: indica que a célula deve ocupar mais de uma coluna ou linha.

E ainda mais funcionalidades II

- `summary`: indica um texto alternativo a ser lido por leitores de tela.
- `scope`: usado em elementos `th` para indicar que é o título de uma linha ou coluna. Mais usado para tabelas complexas.
- Estudar mais em [1].

Referências

[1] Web Education Community Group. HTML tables, 2013. URL http://www.w3.org/community/webed/wiki/HTML_tables.

Parte XIII

Introdução a Formulários em HTML

Introdução

- Um formulário é qualquer área onde o usuário pode inserir informações em uma página.
- Quando o usuário submete o formulário, as informações são enviadas para um servidor web para tratamento.

Elemento form

- `<form>`: indica o começo e fim de um formulário. Formulários não podem ser aninhados.
- Principais atributos:
  - `action`: Quem tratará os dados submetidos pelos formulário. Geralmente arquivos ASP, PHP e JSP.
  - `method`: Método HTTP a ser utilizado no envio de dados. Geralmente GET ou POST.

```
<form action="script.php" method="post">
...
</form>
```

Elemento input I

- `<input>`: elemento mais comum para entrada de dados pelo usuário.
- Principais atributos:
  - `type`: Tipo de entrada de dados. Altera o formato de interação com o usuário.
  - `name`: Nome do campo, de forma a diferenciá-lo dos demais.
  - `value`: Valor padrão do campo. Este atributo receberá o valor informado pelo usuário.

```
<form action="script.php" method="post">
  Nome: <input type="text" name="name" id="name" value="" />
  <input type="submit" value="submit" />
</form>
```

Elemento `input`

Atributo `type`

- `text`
- `submit`
- `radio`
- `checkbox`
- `password`
- `hidden`
- `reset`
- `button`
- `file`
- `image`
- `number`
- `range`
- `color`
- `date`
- `email`
- `search`
- `month`
- `week`
- `tel`
- `datetime`

Elemento `input`
Outros atributos

- `autofocus`
- `checked`
- `disabled`
- `min`
- `max`
- `maxlength`
- `multiple`
- `pattern`
- `placeholder`
- `readonly`
- `required`
- `size`
- `src`
- `step`

Outros elementos de formulário I

- <textarea>: provê uma área de texto com múltiplas linhas. O tamanho deste campo é dado pelos atributos `cols` e `rows`, dados em caracteres.
- <select>: provê uma lista de opções para o usuário escolher. Cada opção é descrita internamente a partir de um elemento <option>. Oferecem resultados semelhantes àqueles dados por uma série de botões radio, embora utilizem menos espaço. De forma geral, listas pequenas de opções mutuamente exclusivas são melhor formatadas como uma série de botões radio.
- <button>: define um botão clicável, similar ao obtido através de <input type="button">.

Outros elementos de formulário II

- `<datalist>`: especifica uma lista de opções pré-definidas para um elemento `<input>`, apresentadas à medida que o usuário digita. Devemos utilizar o atributo `id` referenciando o atributo `list` no `<input>`.

Adicionando semântica, estilo e mais estrutura

- O elemento `<fieldset>` organiza o formulário em módulos semânticos.
- O elemento `<legend>` indica o nome do `<fieldset>`.
- O elemento `<label>` dá uma indicação que relaciona o texto com o campo específico. O atributo `for` deve conter a referência para o campo relacionado.
- O elemento `<output>` representa o resultado de um cálculo. O atributo `for` deve conter a referência para o campo relacionado.
- Estudar mais em [1, 2].

Referências

[1] Web Education Community Group. HTML forms - the basics, 2012. URL http://www.w3.org/community/webed/wiki/HTML_forms_-_the_basics.

[2] Web Education Community Group. HTML5 form additions, 2012. URL http://www.w3.org/community/webed/wiki/HTML5_form_additions.

Parte XIV

Introdução a CSS

Introdução

- Agora que vimos como estruturar uma página usando HTML, estudaremos CSS (*Cascading Style Sheets*), que é usado para estilizar o HTML e posicionar seus elementos na tela.
- Veremos o que CSS é, como aplicá-lo a HTML, e como é sua sintaxe básica.

O que é CSS?

- CSS dá informações ao navegador sobre como apresentar certo elemento – estilizar, espaçamento, e posicionamento.
- Ele usa um sistema de regras, que indicam quais elementos HTML devem possuir um estilo associado, e dentro de cada regra lista as propriedades (cor, tamanho, fonte, etc.) que deseja manipular e para quais valores modificá-los.
- CSS não é uma linguagem de programação, como JavaScript, nem uma linguagem de marcação, como HTML.

Definindo regras de estilo

- O seletor indica o elemento HTML que a regra será aplicada.
- As chaves contém os pares de propriedade/valor, separados por ponto-e-vírgula. As propriedades são separadas dos valores por dois pontos.
- As propriedades definem o que se deseja fazer com o elemento selecionado.
- Os valores indicam para que valor a propriedade deve ser alterada. Depende da propriedade.

Definindo regras de estilo

- Modelo:

```
selector {
  property1: value;
  property2: value;
  property3: value;
}
```

- Exemplo:

```
p {
  margin:5px;
  font-family:arial;
  color:blue;
}
```

Comentários

- Para adicionar comentários devemos encapsular o texto entre /* e */, podendo conter várias linhas.
- Pode aparecer entre regras e entre propriedades. Não existe comando para comentário de linha.

```
/* These are basic element selectors */
selector{
  property1:value;
  property2:value;
  /*property3:value;*/
}
```

Agrupando seletores

- Pode-se agrupar seletores usando vírgula.

```
/* Dois seletores */
h1 {color:red;}
p {color:red;}

/* Igual a */
h1, p {color:red;}
```

- Existem vários tipos de seletores, cada um correspondendo a uma parte distinta da página. Os mais básicos são: seletor de elemento, seletor de classe e seletor de id.

Agrupando seletores

- `p {}`: seletor de elemento
  - seleciona todos os elementos com aquele nome específico.
- `.exemplo {}`: seletor de classe
  - seleciona todos os elementos que possuem o atributo **`class`** com aquele valor específico.
- `#exemplo {}`: seletor de id
  - seleciona todos os elementos que possuem o atributo **`id`** com aquele valor específico.

Agrupando seletores para formar regras mais específicas

- `p.warning {}`: seleciona todos os parágrafos com o atributo `class` igual a `warning`.
- `div#exemplo {}`: seleciona os elementos com o atributo `id` igual a `example`, mas só se for um `div`.
- `p.info, li.highlight {}`: seleciona os parágrafos com `class` igual a `info` e itens de lista com `class` igual a `highlight`.

Seletores avançados
Seletor universal

- Seleciona todos os elementos e aplica estilos a eles.
- Por exemplo, a seguinte regra indica que todo elemento da página deve possuir uma borda sólida de 1 pixel.

```
* {
  border: 1px solid #000000;
}
```

Seletores avançados
Seletor de atributo

- Permite selecionar elementos baseado nos atributos que ele contém. Por exemplo, pode-se selecionar todos os elementos `<img>` com atributo `alt`, usando o seguinte seletor:

```
img[alt] {
  border: 1px solid #000000;
}
```

- Pode-se também selecionar pelo valor do atributo.

```
img[src="alert.gif"] {
  border: 1px solid #000000;
}
```

Seletores avançados
Seletor de filho

- Pode-se usá-lo para selecionar elementos que são filhos de outros elementos. Por exemplo, a regra seguinte torna o texto dos elementos que são filhos de elementos <h3> azul.

```
h3 > strong {
  color: blue;
}
```

Seletores avançados
Seletor de descendentes

- Similares aos seletores de filhos, exceto que estes só selecionam filhos diretos; seletores de descendentes selecionam elementos em qualquer nível da estrutura hierárquica.

```
<div>
  <em>hello</em>
  <p>In this paragraph I will say <em>goodbye</em>.</p>
</div>
```

- `div > em {}` seleciona somente o primeiro `<em>`.
- `div em {}` seleciona os dois elementos `<em>`.

Seletores avançados
Seletor de irmãos adjacentes

- Permitem selecionar um elemento específico que vem diretamente após outro elemento, no mesmo nível hierárquico. Por exemplo, podemos selecionar somente os parágrafos que vem imediatamente depois de elementos <h2>.

```
h2 + p {
   ...
}
```

Seletores avançados
Pseudo-classes

- São usados para prover estilos não para elementos, mas para os vários estados de um elemento. Mais usados para estilizar os estados de links.
  - `:link` – o estado padrão dos links.
  - `:visited` – links já visitados.
  - `:focus` – links (ou campos de formulário, ou qualquer outra coisa) onde o cursor do teclado esteja selecionado.
  - `:hover` – links cujo mouse esteja atualmente apontando.
  - `:active` – um link que está sendo clicado.

Seletores avançados
Pseudo-classes

```
a:link {
  color: blue;
}

a:visited {
  color: gray;
}

a:hover, a:focus {
  text-decoration: none;
}

a:active {
  font-weight: bold;
}
```

Seletores avançados
Pseudo-classes

- `:first-child` – seleciona qualquer instância do elemento que é o primeiro filho. A seguinte regra seleciona o primeiro item de lista (ordenada ou não) de qualquer lista e o deixa em negrito:

```
li:first-child {
  font-weight: bold;
}
```

- `:lang` – seleciona elementos cujo atributo `lang` foi setado para um idioma específico.

```
<p lang="en-US">A paragraph of American text, gee whiz!<p>

p:lang(en-US) { }
```

Seletores avançados I
Pseudo elementos

- `::first-letter` – seleciona a primeira letra de um determinado elemento.

```
p::first-letter {
  font-weight: bold;
  font-size: 300%
  background-color: red;
}
```

- `::first-line` – seleciona a primeira linha de um determinado elemento.

```
p::first-line {
  font-weight: bold;
}
```

- `::before` – usado para acrescentar informações antes do conteúdo do elemento.

Seletores avançados II

Pseudo elementos

```
h1::before {
    content: url(smiley.gif);
}
```

- `::after` – usado para acrescentar informações após do conteúdo do elemento.

```
h1::after {
    content: url(smiley.gif);
}
```

- `::selection` – seleciona a porção de um elemento que é selecionada pelo usuário.

```
::selection {
    color: red;
    background: yellow;
}
```

Aplicando CSS a HTML
Estilos nos elementos

- Existem três formas: estilos nos elementos, estilos embutidos na página e estilos externos.
- Estilos nos elementos são aplicados usando o atributo **style**.

```
<p style="background: blue; color: white; padding: 5px;">Paragraph
    </p>
```

- Força o navegador a usar esses estilos. Qualquer outro estilo será sobrescrito.
- A manutenção fica mais trabalhosa.

Aplicando CSS a HTML
Estilos embutidos na página

- Ficam localizados no cabeçalho do documento, dentro de um elemento <style>.

```
<style type="text/css" media="screen">
  p {
    color: white;
    background: blue;
    padding: 5px;
  }
</style>
```

- Não é preciso aplicar o estilo em cada elemento separadamente, e sim no cabeçalho da página. E se precisarmos modificar o estilo de diversas páginas simultaneamente?

Aplicando CSS a HTML
Estilos externos

- Todos as regras CSS devem estar em um arquivo separado, com extensão .css, e sendo aplicado a página usando o elemento <link> que deve estar no cabeçalho da página.

```
<link rel="stylesheet" href="styles.css" type="text/css" media="
    screen">
```

- Mudanças neste arquivo são aplicadas a todas as páginas que a importam. Uma vez baixados pelo navegador, são armazenados em cache, economizando largura de banda.

Aplicando CSS a HTML
Estilos externos

- Outra forma de importar estilos é usar o comando `@import`, dentro do elemento `<style>`.

```
<style type="text/css" media="screen">
  @import url("styles.css");
  /* ou... */
  @import url("styles.css") screen;

  ...other import statements or CSS styles could go here...
</style>
```

- Deve ser o primeiro comando a aparecer.
- Ver lista completa de seletores em [1].
- Mais sobre esta aula em [2, 3].

Referências

[1] W3Schools.com. CSS Selectors Reference, 2015. URL http://www.w3schools.com/cssref/css_selectors.asp.

[2] Web Education Community Group. CSS basics, 2012. URL http://www.w3.org/community/webed/wiki/CSS_basics.

[3] Web Education Community Group. Advanced CSS selectors, 2013. URL http://www.w3.org/community/webed/wiki/Advanced_CSS_selectors.

Parte XV

Herança e Cascata

Introdução

- Herança está associada em como os elementos de HTML herdam propriedades de seus pais e as repassam para seus filhos, enquanto a cascata tem a ver com declarações CSS sendo aplicadas a um documento, e como regras conflitantes sobrescrevem ou não outras regras.

Herança

- Mecanismo pelo qual elementos repassam suas propriedades para seus filhos.
- Nem todas as propriedades são herdadas; margens, por exemplo.
- Ela é útil pois nos permite aplicar um estilo para um elemento, e o mesmo será aplicado automaticamente para seus filhos. Caso contrário, teríamos que informar o estilo para todos os elementos da página.

Herança

- Considerando a seguinte página HTML.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <link rel="stylesheet" type="text/css" href="style.css">
    <title>Inheritance</title>
  </head>
  <body>
    <h1>Heading</h1>
    <p>A paragraph of text.</p>
  </body>
</html>
```

Herança

- E o documento CSS abaixo, verificamos que mesmo sem aplicar estilo aos elementos internos da página, a aparência deles foi modificada.

```
html {
  font-size: 75%;
  font-family: Verdana, sans-serif;
}
```

- O elemento HTML vai possuir tamanho de 75% em relação a quem? O elemento <body> não deveria receber um valor de 75% do tamanho do elemento pai?

Herança

- O valor a ser herdado não é o valor especificado – o que escrevemos no estilo – mas o chamado *valor computado*.
- O elemento `<html>` vai possuir tamanho de fonte de 75% do tamanho padrão especificado pelo navegador (em geral 16px). 75% de 16 é 12px, e este valor será repassado aos seus filhos.
- Adicionando mais duas declarações ao estilo CSS:

```
html {
  font: 75% Verdana, sans-serif;
  background-color: blue;
  color: white;
}
```

Forçando a herança

- Pode-se forçar a herança – mesmo para propriedades que não são herdadas por padrão – usando a palavra chave `inherit`.
- A regra seguinte faz com que todos os parágrafos herdem as propriedades de fundo de tela de seus pais:

```
p {
  background: inherit;
}
```

A cascata

- É o mecanismo que controla o resultado final quando declarações CSS múltiplas e conflitantes são aplicadas a um mesmo elemento. Existem três conceitos que controlam a ordem na qual as declarações CSS são aplicadas:
  1. Importância
  2. Especificidade
  3. Ordem do código fonte
- A ordem de precedência são as descritas acima.

Importância

- Depende de *onde* ela foi especificada. As declarações conflitantes serão aplicadas na seguinte ordem; as últimas sobrescrevem as primeiras:
  1. Estilos do agente de usuário
  2. Declarações normais nos estilos do usuário
  3. Declarações normais nos estilos do autor
  4. Declarações importantes nos estilos do autor
  5. Declarações importantes nos estilos do usuário
- Declarações importantes são aquelas seguidas pela diretiva `!important`.

```
* {
  font-family: "Comic Sans MS" !important;
}
```

Especificidade

- Mede o quão específica uma regra é. Um seletor com baixa especificidade pode selecionar vários elementos, enquanto um seletor com alta especificidade pode apenas selecionar um único elemento da página.
- Especificidade possui quatro componentes: vamos chamá-los de *a*, *b*, *c* e *d*. Componente *a* é o mais distintivo, *d* o menos.
  1. Componente *a* é bem simples: vale 1 para uma declaração em um atributo `style`, caso contrário é 0.
  2. Componente *b* é o número de seletores `id`.
  3. Componente *c* é o número de seletores de atributos – incluindo seletores de classe – e pseudo-classes.
  4. Componente *d* é o número de tipos de elementos e pseudo-elementos no seletor.

Especificidade

Seletor	a	b	c	d	Especificidade
`h1`				1	0,0,0,1
`.foo`			1		0,0,1,0
`li:first-child h2 .title`	0	0	2	2	0,0,2,2
`#bar`		1			0,1,0,0
`#nav .selected > a:hover`	0	1	2	1	0,1,2,1
`html>head+body ul#nav *.home a:link`		1	2	5	0,1,2,5

- Calculador automático de especificidade [1].

Ordem do código fonte

- Se duas declarações afetam o mesmo elemento, possuem a mesma importância e a mesma especificidade, a distinção final será dada pela ordem no código fonte. A declaração que aparece por último nos estilos possui precedência sobre as que vem antes.
- Mais sobre esta aula em [2].

Referências

[1] K. Street. Specificity Calculator, 2015. URL http://specificity.keegan.st/.

[2] Web Education Community Group. Inheritance and cascade, 2012. URL http://www.w3.org/community/webed/wiki/Inheritance_and_cascade.

Parte XVI

Estilizando texto em CSS

Propriedades de fonte em CSS

Cor da fonte

- A propriedade primária de interesse é `color`. Pode ser definida através do(a):
  - Nome da cor. Valores possíveis:
    `aqua`, `black`, `blue`, `fuchsia`, `gray`, `green`, `lime`, `maroon`, `navy`, `olive`, `orange`, `purple`, `red`, `silver`, `teal`, `white`, `yellow`.
  - Notação hexadecimal. Exemplos: `#f00`, `#abcdef`, `#a1b23c`, `#def`.
  - Notação funcional. Exemplos:
    `rgb(50,100,150)`, `rgb(10%, 20%, 30%)`. Proibido misturar números e porcentagens.
  - Notação funcional estendida (opacidade). Exemplos:
    `rgba(50,100,150, 0.5)`, `rgba(10%, 20%, 30%, 0.8)`.

Propriedades de fonte em CSS
Tamanho da fonte

- A propriedade primária de interesse é **font-size**. Quando usado em uma regra, é seguida por um valor que especifica a unidade de medida, ou algumas vezes por uma palavra chave (como **small** ou **medium**).

```
body {
  font-size: 14px;
}
```

Propriedades de fonte em CSS

Tamanho da fonte

- Tamanhos absolutos são melhor usados em layouts que não mudam muito em relação às propriedades da tela.
- Tamanhos relativos devem ser usados em layouts não estáticos, e em situações em que um acordo deve ser feito entre usabilidade e controle do layout pelo design.
- Tamanhos com palavras chave devem ser usadas quando a usabilidade é mais importante que qualquer outra consideração de design.

Tipo	**Valores**
Relativos	em, ex, percentual
Absolutos	px, pt, pc, in, cm
Nomeados	xx-small, x-small, small, medium, large, x-large, xx-large

Propriedades de fonte em CSS

Família da fonte

- `font-family` indica o tipo de fonte a ser usado. Deve-se seguir as seguintes regras:
  - O nome deve ser idêntico ao disponível na biblioteca do computador.
  - Todos os nomes devem ser separados por vírgula.
  - Se o nome contiver mais de uma palavra, deve-se usar aspas simples ou duplas. Exemplo: 'Times New Roman'.
  - Devem ser escritas em ordem crescente de disponibilidade.
  - Deve-se sempre terminar a lista com um nome de família genérico.

```
body {
  font-family: Palatino, 'Palatino Linotype', Georgia, sans-serif;
}
```

Propriedades de fonte em CSS

Modificando os detalhes

- `font-style` manipula o itálico. Possui três valores válidos: `italic`, `oblique`, e `normal`.
- `font-variant` possui dois valores: `small-caps` e `normal`.
- `font-weight` possui vários valores. Os mais comuns são: `bold` e `normal`.

```
em {
  font-size: large;
  font-style: normal;
}

strong {
  font-variant: small-caps;
  font-weight: normal;
  font-style: italic;
}
```

Propriedades de fonte em CSS
A propriedade minimizada

- Permite resumir várias propriedades em uma única.

```
h1 { font: italic normal bold x-large/1.167em Helvetica, Arial,
     sans-serif; }
```

- Deve-se prover valores para todos as propriedades na seguinte ordem.
  - **font-style**
  - **font-variant**
  - **font-weight**
  - **font-size**, seguido se necessário por uma barra e o valor de **line-height**
  - **font-family**

Sombreamento

- A propriedade **box-shadow** acrescenta uma ou mais sombras à um elemento. A propriedade é uma lista de sombras separadas por vírgulas, cada uma especificada por 2 a 4 valores de tamanho, uma cor opcional, e uma palavra chave opcional **inset**.
- Os componentes desse comando são interpretados da seguinte forma:
  - O primeiro tamanho é o espaçamento horizontal da sombra.
  - O segundo tamanho é o espaçamento vertical da sombra.
  - O terceiro tamanho é a mancha da sombra.
  - O quarto tamanho é o espalhamento da sombra.
  - A cor da sombra.
  - A palavra **inset**, se presente, cria uma sombra interna ao elemento.

```
div {
  box-shadow: 10px 10px 5px #888888;
}
```

Propriedades de fonte em CSS

Propriedade	Valores
font-family	cursive, fantasy, monospace, sans-serif, serif
font-size	xx-small, x-small, small, medium, large, x-large, xx-large
font-style	italic, oblique, normal
font-variant	small-caps, normal
font-weight	bold, normal
line-height	normal
text-align	left, right, center, justify
text-decoration	line-through, none, overline, underline
text-transform	capitalize, lowercase, none, uppercase
white-space	normal, nowrap, pre, pre-line, pre-wrap

- Mais sobre esta aula em [1].

Referências

[1] Web Education Community Group. CSS text styling part 1, 2012. URL http://www.w3.org/community/webed/wiki/CSS_text_styling_part_1.

Parte XVII

Modelo de Layout em CSS

Introdução

- Veremos as propriedades CSS que manipulam o layout dos elementos HTML, incluindo bordas, margens, dentre outras.

Margens

- As margens podem ser especificadas uma a uma, ou usando a forma simplificada.
- As margens só são aplicáveis a elementos de bloco (exceto imagens).
- Podemos aplicar às margens valores em unidades absolutas ou relativas.
- As propriedades para setar as margens: **`margin-top`**, **`margin-right`**, **`margin-bottom`**, **`margin-left`**.

Margens

- Podemos usar a forma simplificada `margin`. Ela permite combinar várias propriedades em uma única propriedade, economizando tempo e esforço.
  1. O mesmo valor aplicado às quatro margens: `margin: 2px;`
  2. Primeiro valor aplicado acima e abaixo, e segundo para esquerda e direita: `margin: 2px 5px;`
  3. Primeiro e terceiro valores aplicados para acima e abaixo respectivamente, segundo valor aplicado à esquerda e direita:
     ```
     margin: 2px 5px 1px;
     ```
  4. Valores aplicados acima, direita, abaixo, e esquerda respectivamente:
     ```
     margin: 1em 1.5em 2em 2.5em;
     ```

Margens

- Margens automáticas:
  - O valor `auto` instrui o navegador para renderizar a margem de acordo com o valor do seu estilo. Entretanto, se tal margem é aplicada a um elemento com uma largura associada, uma margem automática faz com que todo o espaço disponível seja apresentado como espaço em branco.
- Margens negativas:
  - Todas as margens podem possuir valores negativos. Quando isso ocorre, uma margem adjacente pode ser efetivamente cancelada em qualquer grau. Aplicando-se uma margem negativa larga suficiente, o elemento adjacente pode ser sobreposta.

Margens

- Margens colapsadas
  - Em casos onde dois elementos de bloco adjacentes e similares compartilham margens maiores que zero, apenas a maior margem será aplicada.
  - `p { margin: 1em auto 1.5em auto; }`
  - Se o documento com este estilo fosse ser renderizado de forma literal, as margens resultantes entre dois parágrafos em série seria de 2.5em, como a soma da margem de baixo do parágrafo 1 (1.5em) e a margem de cima do parágrafo 2 (1em). Entretanto, por causa da aplicação das margens colapsadas, a margem será de 1.5em.

Bordas

- Podemos informar a largura, estilo, e cor de qualquer uma das quatro bordas. As propriedades são:

  - `border-width`
  - `border-style`
  - `border-color`
  - `border-top`
  - `border-top-width`
  - `border-top-style`
  - `border-top-color`
  - `border-right`
  - `border-right-width`
  - `border-right-style`
  - `border-right-color`
  - `border-bottom`
  - `border-bottom-width`
  - `border-bottom-style`
  - `border-bottom-color`
  - `border-left`
  - `border-left-width`
  - `border-left-style`
  - `border-left-color`

Bordas

- Todas suportam uma versão simplificada, como as margens.
- `border-width` atribui a largura das bordas. Percentuais não são aceitos. Exemplo: `td { border-width: 1px 0 0 1px; }`
- `border-style` aceita um dos seguintes valores: `dashed`, `dotted`, `double`, `inset`, `groove`, `outset`, `ridge`, `solid`, `none`.
- `border-color` aceita uma cor que será atribuída a uma borda.
- `border` é a versão simplificada que setará os valores para as quatro bordas. A ordem é: largura, estilo e cor. Exemplo:

```
border: 2px outset rgb(160,0,0);
```

Bordas arredondadas I

- Com CSS3 podemos criar bordas arredondadas, sombras e usar uma imagem como borda.
- Para criar bordas arredondadas usaremos a propriedade **border-radius**. Informando apenas um valor, todas as bordas são afetadas.

```
div {
  border: 2px solid #a1a1a1;
  padding: 10px 40px;
  background: #dddddd;
  width: 300px;
  border-radius: 25px;
}
```

- Este exemplo funciona da mesma forma que:

Bordas arredondadas II

```
div {
  border: 2px solid #a1a1a1;
  padding: 10px 40px;
  background: #dddddd;
  width: 300px;
  border-top-left-radius: 25em;
  border-top-right-radius: 25em;
  border-bottom-right-radius: 25em;
  border-bottom-left-radius: 25em;
}
```

- Informando dois valores, as bordas (top,left) e (bottom,right) são afetadas.
- Informando três valores, o primeiro valor afeta a borda (top,left). O segundo valor afeta as bordas (top,right) e (bottom,left). O terceiro valor afeta a borda (bottom,right).
- Informando quatro valores, as bordas são afetadas na seguinte ordem: (top,left), (top,right), (bottom,right) e (bottom,left).

Bordas arredondadas III

- As formas acima atribuem o mesmo valor aos raios horizontais e verticais. Outra forma de alterar as bordas, detalhando os raios horizontais e verticais é informá-los separados por uma barra. Por exemplo:

```
border-radius: 2em 1em 4em / 0.5em 3em;
```

- É o mesmo que:

```
border-top-left-radius: 2em 0.5em;
border-top-right-radius: 1em 3em;
border-bottom-right-radius: 4em 0.5em;
border-bottom-left-radius: 1em 3em;
```

Padding

- Espaçamento entre o conteúdo e as bordas. Podemos usar as propriedades **`padding`**, **`padding-top`**, **`padding-right`**, **`padding-bottom`**, **`padding-left`**.
- Funcionam da mesma forma que as margens, com as seguintes exceções:
  - Valores **`auto`** são funcionalmente inúteis.
  - Valores negativos são inválidos.
  - **`padding`** nunca é colapsado.
  - Valores de margens não são aplicados a elementos de linha, mas valores de **`padding`** são.

Trabalhando com altura e largura

- A maioria dos elementos pode ter suas dimensões alteradas.
- As propriedades que permitem a manipulação das dimensões são: **`width`**, **`min-width`**, **`max-width`**, **`height`**, **`min-height`**, **`max-height`**.
- Cuidados a serem tomados:
  - **`width`** e **`height`** não podem ser aplicados a elementos de linha, exceto para imagens.
  - **`width`** e **`height`** são apenas duas das propriedades que influenciam as dimensões de um elemento.
  - Algoritmos de arredondamento podem causar diferenças no layout entre navegadores que apresentam conteúdo via mídias do tipo LCD, LED, ou CRT (`type="screen"`).

Trabalhando com altura e largura I

Exemplo

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title>Exemplo</title>
<style>
#box { width: 50%; min-width: 200px; max-width: 400px; background-color
    : yellow; }
#min { width: 200px; background-color: red; }
#max { width: 400px; background-color: blue; }
</style>
</head>
<body>
<p id="min">Minimum width</p>
```

Trabalhando com altura e largura II

Exemplo

```
<p id="box">The maximum width of this paragraph is set to 100px. The
    maximum width of this paragraph is set to 100px. The maximum width
    of this paragraph is set to 100px. The maximum width of this
    paragraph is set to 100px. The maximum width of this paragraph is
    set to 100px.</p>
<p id="max">Maximum width</p>
</body>
</html>
```

Sobreposição

- Podemos usar a propriedade `overflow` e seus quatro valores válidos – `visible`, `hidden`, `auto`, `scroll` – para indicar o comportamento de um elemento quando seu conteúdo é maior que o tamanho do elemento.
  - `visible` – Conteúdo além das dimensões disponíveis de um elemento são apresentadas sem afetar o fluxo ou margens de elementos adjacentes. Conteúdo de um elemento pode parecer colidir com conteúdo de seus vizinhos.
  - `hidden` – Qualquer conteúdo que fique além dos limites de um elemento serão escondidos.
  - `auto` – As dimensões de um elemento serão restringidas da mesma forma quando o valor `hidden` é usado, exceto que as barras de rolagem serão criadas se necessário para tornar a leitura do conteúdo acessível.
  - `scroll` – Barras de rolagem horizontal e vertical serão incorporadas ao elemento, mesmo se não forem necessários.

O Modelo de Caixa de CSS

- Agora que aprendemos as propriedades básicas de layout, vejamos como a largura é apresentada de acordo com as propriedades CSS.
- Da mesma forma que texto, elementos podem ser dimensionados usando unidades proporcionais ou estáticas.
- Primeira regra ao modificar o tamanho de elementos: misture unidades proporcionais e estáticas com cuidado ou então não faça.
- O valor padrão de `width` é `auto` (ocupe o espaço disponível) e de `height` é englobar o conteúdo.

O Modelo de Caixa do W3C

- A regra básica é que a largura ou altura computadas é igual a:

```
margin + border + padding + (width|height)
```

- Considerando a regra abaixo:

```
#myLayoutColumn {
  width: 50em;
  margin: 1.5em auto 1.5em auto;
  border: .1em;
  padding: .9em;
}
```

- A largura não-marginal do elemento será de: .1em + .9em + 50em + .9em + .1em = 52em

Trabalhando com o fluxo do documento

Display

- Vejamos algumas propriedades que afetam o fluxo do documento: `display`, `float`, e `clear`.
- Com a exceção de `html`, `body`, e `table`, todos os elementos com associação a conteúdo são dos tipos bloco ou linha.
- A propriedade `display` possui três valores mais comuns: `block`, `inline`, e `none`.
- Com eles, podemos alterar o comportamento padrão dos elementos.

Trabalhando com o fluxo do documento

Flow

- Permite informar como elementos subsequentes devem flutuar em relação a um elemento. A propriedade que permite isso é **`float`**. Os valores mais comuns são: **`left`**, **`right`**, e **`none`**.
  - Um valor de **`float`** só será aplicado se o elemento for de bloco com uma largura explicita.
  - As propriedades **`float`**, **`clear`**, e **`margin`** aparecem juntas para criação de colunas em um layout.

Trabalhando com o fluxo do documento

Clear

- A propriedade `clear` informa como um elemento deve flutuar em relação a todos os seus vizinhos.
- Permite os valores `left`, `right`, `none`, ou `both`.
- Mais sobre esta aula em [1].

Referências

[1] Web Education Community Group. The CSS layout model - boxes borders margins padding, 2011. URL http://www.w3.org/community/webed/wiki/The_CSS_layout_model_-_boxes_borders_margins_padding.

Parte XVIII

Imagens de Fundo em CSS

Plano de fundo
Cor

- Vejamos as propriedades utilizadas para modificar o plano de fundo de um site web.
- `background-color` define a cor de fundo do site. Utiliza notação de cores apresentada previamente.
- Outros valores válidos são `transparent` e `inherit`.

Plano de fundo
Imagem e Repetição

- **background-image** indica o caminho ou URL da imagem de fundo. Exemplo: `background-image: url(alert.png);`
- Outros valores válidos são **none** e **inherit**.
- A propriedade **background-image** aceita múltiplas camadas de imagens de fundo através de uma lista de figuras, onde a primeira aparecerá mais próxima do usuário.

```
div {
  background-image:url(img_flwr.gif),url(img_tree.gif);
}
```

- **background-repeat** indica para qual direção a imagem deve ser repetida. Elas podem ser repetidas horizontalmente, verticalmente, ou ambas, para preencher toda a largura ou altura de um elemento.
- Os valores válidos incluem **repeat**, **repeat-x**, **repeat-y**, e **no-repeat**.

- `background-attachment` indica o comportamento da imagem de fundo quando o usuário usa as barras de navegação.
- Os valores válidos incluem `scroll`, `fixed`, e `inherit`.
- `background-position` informa ao navegador onde posicionar a imagem de fundo.
- Imagens podem ser posicionadas em qualquer lugar dentro das bordas do elemento. Podemos indicar a posição usando palavras-chave ou valores numéricos.
- Composto de dois valores: espaçamento horizontal seguido de espaçamento vertical.

Plano de fundo
Propriedade minimizada

- A propriedade minimizada **background** permite informar todas as propriedades acima em uma linha.
- A ordem deve ser a seguinte:
  1. color
  2. image
  3. repeat
  4. attachment (pode ser omitida)
  5. posição horizontal
  6. posição vertical
- Exemplo: `background: green url(logo.gif)no-repeat left top;`

Plano de Fundo I

- **background-size** informa o tamanho da imagem de fundo.

```
div {
  background: url(img_flwr.gif);
  background-size: 80px 60px;
  background-repeat: no-repeat;
}
```

- **background-origin** especifica a área de posicionamento das imagens de fundo. Os valores possíveis são **content-box**, **padding-box** ou **border-box**.

```
div {
  background: url(img_flwr.gif);
  background-repeat: no-repeat;
  background-size: 100% 100%;
  background-origin: content-box;
}
```

Plano de Fundo II

- A propriedade **background-clip** especifica a área de pintura da cor de fundo.

```
div {
  padding: 25px;
  border: 5px dotted #000000;
  background-color: yellow;
  background-clip: content-box;
}
```

- Mais sobre esta aula em [1].

Referências

[1] Web Education Community Group. CSS background images, 2011. URL http://www.w3.org/community/webed/wiki/CSS_background_images.

Parte XIX

Estilizando Listas e Links em CSS

Introdução

- Podemos escolher que forma de numeração ou marcação usar para listas. Para isso, usamos a propriedade **`list-style-type`**.
- A cor da marcação ou números é a mesma usada pelo elemento <**`li`**>. Caso se deseje cores diferentes, deve-se usar imagens.

```
ul li {
  list-style-type: square;
}
```

Marcadores personalizados I

- Podemos usar imagens ao invés dos marcadores padrão. Para isso, usamos a propriedade **list-style-image**.

```
ul {
  list-style-image: url("http://png.com/ellipse.png")
}
```

- Outra forma é remover o marcador padrão e adicionar uma imagem de fundo para cada item.
- Considerando a lista abaixo:

```
<ul class="rss">
  <li><a href="http://example.com/rss.xml">News</a></li>
  <li><a href="http://example.com/rss.xml">Sport</a></li>
  <li><a href="http://example.com/rss.xml">Weather</a></li>
  <li><a href="http://example.com/rss.xml">Business</a></li>
  <li><a href="http://example.com/rss.xml">Entertainment</a></li>
  <li><a href="http://example.com/rss.xml">Funny News</a></li>
</ul>
```

Marcadores personalizados II

- Podemos estilizá-la usando o seguinte código CSS:

```
.rss {
  margin: 0;
  padding: 0;
  list-style-type: none;
}

.rss li {
  background: #fff url("icon-rssfeed.gif") 0 3px no-repeat;
  padding: 0 0 5px 15px;
}
```

Posição dos marcadores

- Se você desejar que o texto de um item com várias linhas fique abaixo do marcador, você pode usar a propriedade **`list-style-position`** para o valor **`inside`**. O padrão é o valor **`outside`**.
- A propriedade minimizada **`list-style`** permite setar os valores das propriedades **`list-style-type`**, **`list-style-image`**, e **`list-style-position`** simultaneamente.

```
ul {
  list-style: disc url("http://png.com/ellipse.png") inside;
}
```

Listas Horizontais

- Uma mudança comum a ser realizada em uma lista é produzir uma lista horizontal.
- Para isso, precisamos fazer três coisas:
  - Remover **margin** e **padding** do elemento
  - Setar os itens da lista para **display: inline;**
  - Dar aos itens da lista algum espaçamento à direita, evitando que eles fiquem colados

```
#mainmenu {
  margin: 0;
  padding: 0;
}

#mainmenu li {
  display: inline;
  padding: 0 1em 0 0;
}
```

Colunas Falsas

- Outra possibilidade visual é apresentar os elementos da lista em duas colunas.

```
.rss {
  margin: 5px 5px 0 5px;
  padding: 0;
  width: 100%;
}

.rss li {
  display: inline-block;
  width: 40%;
  margin: 0 2% 0 0;
  list-style-type: none;
  background: #fff url("icon-rssfeed.gif") 0 3px no-repeat;
  padding: 0 0 5px 15px;
}
```

Estilizando links

- Para estilizar links e obter bons resultados é necessário ter algumas regras em mente:
  - entender os diferentes estados que um link pode se encontrar
  - não ficar muito distante das expectativas dos usuários
  - usar cores com cuidado

Estados de um link

- Um link pode se encontrar em um de cinco estados diferentes:
  - `:link` – O estado padrão de um link quando ele não foi ativado ou visitado previamente.
  - `:visited` – Quando o usuário já visitou este link.
  - `:focus` – Aplicado quando o link possui o foco. Por exemplo, quando o cursor do teclado está sob o link.
  - `:hover` – Aplicado enquanto o usuário passa o mouse sob o link, mas ainda não o clicou.
  - `:active` – Aplicado enquanto o usuário ativa o link (literalmente durante o tempo que ele o está clicando).

Estilizando links na ordem correta

- Deve-se estar ciente que se não colocar os estilos dos links na ordem correta, as propriedades vão sobrescrever umas às outras e os estados não funcionarão. A ordem correta é a seguinte:
  1. `link`
  2. `visited`
  3. `focus`
  4. `hover`
  5. `active`
- Caso se deseje aplicar um estilo a todos os links, independentemente do estado, este deve aparecer antes de todos os outros apresentados acima.
- Mais sobre esta aula em [1].

Referências

[1] Web Education Community Group. Styling lists and links, 2011. URL http://www.w3.org/community/webed/wiki/Styling_lists_and_links.

Parte XX

Estilizando Tabelas em CSS

Introdução

- Tabelas não devem ser usadas para layout. Veremos como posicionar elementos na tela nas próximas aulas.
- Os elementos estruturais chave que precisaremos estilizar são:
  - Cabeçalhos
  - Células de dados
  - Título da tabela
- Para facilitar a interpretação, devemos garantir que estes elementos sejam claramente diferentes. As formas mais comuns são pela utilização de bordas, cores de fundo, ou ambas.

Largura da tabela e das células

- A primeira decisão é quão larga a tabela deve ser.
- Por padrão, as tabelas ocuparão toda a largura disponível. Podemos alterar a largura da tabela através da propriedade **`width`**.

```
table {
  width: 100%;
}

th, td {
  width: 25%;
}
```

Alinhamento

- Podemos ajustar o alinhamento horizontal através da propriedade **text-align**. O alinhamento vertical pode ser ajustado através de **vertical-align**.

```
table {
  width: 100%;
}

th, td {
  width: 25%;
  text-align: left;
  vertical-align: top;

}
```

Bordas

- É necessário setar as bordas separadamente para cada parte da tabela, e então decidir como as bordas devem combinar.
- A borda da tabela fica por fora das células de cabeçalho e dados.

```
table {
  width: 100%;
  border: 1px solid #000;
}

th, td {
  width: 25%;
  text-align: left;
  vertical-align: top;
  border: 1px solid #000;
}
```

Bordas

- Podemos remover o espaçamento entre as bordas de duas formas: através da propriedade **border-spacing** ou **border-collapse**.
- **border-spacing** pode receber um valor a ser aplicado às quatro bordas, ou dois valores, o primeiro para espaçamento horizontal e o segundo, vertical.
- **border-collapse** pode receber dois valores: **separate**, que mantém as bordas separadas; e, **collapse**, que une as bordas de células vizinhas.

```
th, td {
  width: 25%;
  text-align: left;
  vertical-align: top;
  border: 1px solid #000;
  border-collapse: collapse;
}
```

Espaçamento e Título da tabela

- Para alterar o espaçamento do texto para as bordas, podemos usar **padding**.
- Podemos também modificar a posição do título da tabela usando **caption-side**. Ele pode ficar acima ou abaixo da tabela.

```
th, td {
  width: 25%;
  text-align: left;
  vertical-align: top;
  border: 1px solid #000;
  border-collapse: collapse;
  padding: 0.3em;
  caption-side: bottom;
}
```

Fundo de tela

- Podemos usar a propriedade `background`, embora você precise estar ciente que as diferentes partes da tabela irão se sobrepor.
- De forma resumida, os fundos de tela irão se sobrepor da seguinte forma:
  1. tabela (que fica ao fundo)
  2. grupos de colunas
  3. colunas
  4. grupos de linhas
  5. linhas
  6. células (topo, significando que seu fundo de tela se sobrepõe a todos os outros)
- Se as bordas estão setadas para `collapse`, o fundo de tela não aparecerá. Caso esteja setado para `separate`, o fundo aparecerá entre as bordas.
- Mais sobre esta aula em [1, 2].

Referências

[1] Web Education Community Group. Styling tables, 2011. URL http://www.w3.org/community/webed/wiki/Styling_tables.

[2] Web Education Community Group. Styling forms, 2011. URL http://www.w3.org/community/webed/wiki/Styling_forms.

Parte XXI

Floats e Clear em CSS

Para que servem?

- Em revistas, geralmente existem imagens ilustrando o conteúdo, com o texto flutuando sobre elas. A propriedade `float` foi criada para permitir esse tipo de layout nas páginas.
- Flutuar uma imagem – ou outro elemento – empurra ela para um lado e deixa o texto fluir pelo outro lado.
- A propriedade `clear` permite que um elemento seja jogado para baixo, evitando que ele apareça ao lado de um elemento flutuante.

Um pouco de teoria

- Cada elemento gera uma caixa onde ele é apresentado. Elementos de bloco são apresentados de cima para baixo. Elementos de linha são apresentados da esquerda para a direita (padrão).
- Isto é chamado *fluxo do documento*.

Como a flutuação funciona?

- A propriedade `float` possui quatro valores válidos: `left`, `right`, `none`, `inherit`.
- Aplicar `float` a um elemento de linha o converte em um elemento de bloco.
- Uma caixa flutuante é removida do fluxo do documento e movida o máximo possível para esquerda ou direita.
- Por exemplo, para `float: left` a caixa é movida para a esquerda até a margem esquerda do elemento flutuante toque a borda esquerda do elemento pai.

Clear

- Garante que um elemento aparecerá sempre após um elemento flutuante.
- A propriedade `clear` possui os valores válidos: `left`, `right`, `none`, `inherit`, e `both`.
- Usando `clear: left` em um elemento significa que ele aparecerá abaixo de qualquer elemento anterior flutuando à esquerda. Se você usar `clear: both` ele aparecerá abaixo de todos os elementos flutuantes de ambos os lados.

Clear

- O elemento pai não se expande para conter os filhos flutuantes. Se todos os elementos estiverem flutuando, o pai terá altura zero.
- Como fazer o pai expandir para conter seus filhos flutuantes?
  1. Flutuar o pai também. Elementos flutuantes sempre se expandem para englobar seus elementos filhos flutuantes.
  2. Setar a propriedade `overflow` do pai para um valor diferente de `visible`. Se setá-los para `hidden` e não especificar uma altura, o pai englobará os filhos flutuantes.

Centralizando

- Um elemento de bloco que sofre flutuação e não possui largura associada, irá reduzir seu tamanho para englobar o seu conteúdo.
- Usando posicionamento de elementos, podemos fazer um elemento flutuante ficar centralizado na tela.
- Para isso modificaremos a posição do elemento para 50% à esquerda e seu pai para -50%.
- Mais sobre esta aula em [1].

Referências

[1] Web Education Community Group. Floats and clearing, 2012. URL http://www.w3.org/community/webed/wiki/Floats_and_clearing.

Parte XXII

Posicionamento em CSS

Para que serve?

- Veremos como posicionar elementos HTML no local que desejarmos na página.
- Usaremos a propriedade CSS `position` que possui quatro valores válidos: `static`, `relative`, `absolute` e `fixed`.

Estático

- Valor padrão.
- Caixas são posicionadas na ordem que elas se encontram no código fonte, ocupando o máximo de espaço necessário.
- Existem diferenças com relação ao tipo de caixa: bloco ou de linha.

Bloco

- Caixas de bloco são posicionadas verticalmente de cima para baixo na ordem que ocorrem no código. Normalmente ocupam a largura do documento. Mesmo se a reduzirmos, os elementos ainda ficarão uma embaixo da outra.
- Uma caixa de bloco conterá ou outras caixas de bloco ou caixas de linha. Caso ocorra uma mistura (erro semântico), caixas anônimas serão criadas para englobar as caixas de linha.

Linha

- Caixas de linha são posicionadas horizontalmente na ordem que ocorrem no código.
- Usando a propriedade `direction` podemos informar se o texto será apresentado da esquerda para a direita (`ltr`) ou da direita para a esquerda (`rtl`).
- Conjunto de caixas de linha que fazem parte de uma linha de texto são englobados em outra caixa, chamada *line box*. Elas não possuem espaçamento entre elas, podendo ser setado pela propriedade `line-height`.
- Não podemos modificar suas dimensões, apenas seus espaçamentos horizontais.
- Podemos alterar seu posicionamento vertical através de `vertical-align` com os valores `baseline`, `bottom`, `middle` e `top`.

Relativo

- Mais relacionado com o posicionamento estático que com o absoluto e fixo.
- É posicionado de acordo com o posicionamento estático, mas a caixa é afetada pelas propriedades `top`, `bottom`, `left` e `right`.
- Apenas a caixa muda de posição. O elemento permanece no mesmo local do posicionamento estático.
- Com relação ao fluxo do documento, o elemento não se moveu – apenas o resultado visual foi movido.
- As propriedades são aplicadas em relação às margens do elemento. É irrelevante aplicar as propriedades `top` e `bottom` (bem como `left` e `right`) simultaneamente. As regras de CSS dizem para ignorar `bottom` se `top` já estiver especificado. Para movimentação horizontal, depende da direção do texto. Se for da esquerda para a direita `right` é ignorado se tanto `right` e `left` estiverem especificados; se for da direita para a esquerda `left` é ignorado.

Absoluto I

- Para estudarmos posicionamento absoluto, temos que conhecer o *containing block*: o caixa de bloco cujas posição e dimensões a caixa com posicionamento absoluto estão relacionadas.
- Nos posicionamentos estático e relativo, o *containing block* é o elemento pai. Para o posicionamento absoluto, é o ancestral posicionado mais próximo (com a propriedade `position` setada para `relative`, `absolute` ou `fixed`). Caso nenhum seja encontrado, o elemento `html` é usado.
- Usaremos as propriedades propriedades `top`, `bottom`, `left` e `right` apresentadas previamente para posicionar a caixa.
- Essas propriedades são relativas ao *containing block*, podendo todas ser utilizadas em conjunto.

Absoluto II

- Elementos posicionados de forma absoluta ocupam somente o tamanho para englobar seu conteúdo. Para alterar suas dimensões devemos usar as propriedades `top` e `bottom` ou `width` para a largura, e `left` e `right` ou `height` para a altura.
- Percentuais são relativos ao *containing block*.

Três dimensões I

- CSS funciona em três dimensões: eixo x é o eixo horizontal, eixo y é o eixo vertical, o eixo z é o eixo da profundidade. Valores de z maiores indicam "na frente" de valores menores.
- Elementos posicionados são apresentados em um *contexto de pilha*.
- Podemos alterar a profundidade através da propriedade `z-index`. Só podemos alterar o índice no eixo z. `z-index` positivo indica à frente de outros elementos dentro do mesmo contexto de pilha.
- Se dois elementos na mesma pilha possuem o mesmo índice, o último a aparecer o código fonte fica à frente.
- Cada pilha contém pelo menos 7 camadas:
  1. Fundo de tela e bordas dos elementos que formam o contexto da pilha.
  2. Descendentes posicionados com nível de pilha negativo.
  3. Descendentes de nível de bloco no fluxo normal.
  4. Descendentes flutuantes
  5. Descendentes de nível de linha no fluxo normal

Três dimensões II

6. Descendentes posicionados com nível de pilha `auto` ou 0 (zero).
7. Descendentes posicionados com nível de pilha positivo.

Fixo

- Elemento é posicionado em relação ao espaço de visão (ao *containing block* inicial). Em geral, a janela do navegador ou as bordas do papel.
- Mais sobre esta aula em [1, 2].

Referências

[1] Web Education Community Group. CSS absolute and fixed positioning, 2011. URL http://www.w3.org/community/webed/wiki/CSS_absolute_and_fixed_positioning.

[2] Web Education Community Group. CSS static and relative positioning, 2013. URL http://www.w3.org/community/webed/wiki/CSS_static_and_relative_positioning.

Parte XXIII

Media Queries

Introdução I

- Consiste de um tipo de mídia e uma expressão que limita o escopo da folha de estilo através do uso de características da mídia, como altura, largura e cor.
- Permite que o conteúdo seja ajustado a dispositivos distintos sem alterar o conteúdo.

```
<!-- CSS media query em um elemento link -->
<link rel="stylesheet" media="(max-width: 800px)" href="example.
    css" />

<!-- CSS media query dentro de uma folha de estilo -->
<style>
@media (max-width: 600px) {
  .facet_sidebar {
    display: none;
  }
}
</style>
```

Introdução II

- Expressões devem estar entre parênteses.
- Tipos de mídia: `all`, `print`, `screen`, `speech`.

Operadores lógicos I

- `and`: combina múltiplas características de um dispositivo em uma única pesquisa.

```
@media (min-width: 700px) and (orientation: landscape) { ... }
```

- `not`: usada para negar uma media query completamente. Deve especificar um tipo de mídia explícito.

```
@media not (all and (monochrome)) { ... }
```

- `only`: usado para aplicar um estilo apenas se toda a pesquisa for satisfeita, útil para evitar que navegadores antigos apliquem os estilos. Deve especificar um tipo de mídia explícito.

```
<link rel="stylesheet" media="only screen and (color)" href="
    example.css" />
```

Operadores lógicos II

- `comma`: permite combinar múltiplas pesquisas em uma lista separada por vírgulas; se qualquer uma for dervadeira, todo o comando é verdadeiro. Equivalente ao operador "ou".

```
@media (min-width: 700px), handheld and (orientation: landscape) {
      ... }
```

Características da mídia

- **`width`**: permite mínimo e máximo.

```
@media screen and (min-width: 500px) and (max-width: 800px) { ...
    }
```

- **`height`**: permite mínimo e máximo.
- `aspect-ratio`: razão entre largura e altura.

```
@media screen and (min-aspect-ratio: 1/1) { ... }
```

- `orientation`: `landscape` ou `portrait`.

```
@media all and (orientation: portrait) { ... }
```

- Mais sobre esta aula em [1, 2].

Referências

[1] Mozilla Developer Network. CSS media queries, 2015. URL https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Media_queries.

[2] W3C. Media Queries, 2012. URL http://www.w3.org/TR/css3-mediaqueries/.

Parte XXIV

Introdução a Acessibilidade

O que é acessibilidade?

- Accessibilidade é tratar todos, independentemente de suas capacidades, da mesma maneira.
- Para desenvolvimento web, isso significa que você precisa tentar estar mais atento das necessidades da audiência que pode visitar seu site.
- Isso pode requerer que você aprenda sobre os diferentes níveis de capacidade que as pessoas podem ter, e como eles usam o computador.

Objetivos

- Aplicando as técnicas apresentadas aqui, você poderá criar sites que funcionam com várias formas de interação.
- Seus sites poderão ser usados por pessoas independentemente se:
  - São cegas ou possuem problemas graves de visão, e ouvem os sites através de leitores de tela, ou os sentem em dispositivos braille.
  - São míopes, e usam fontes grandes.
  - Possuem problemas motores que impossibilitem de usar as mãos para manipular um mouse, e usam um dispositivo apontador para usar o teclado, ou um apontador de olhos para manipular o site.
  - Usa trackballs, ou outra forma não usual para controlar o computador.

Por que a acessibilidade é importante?

- A principal razão é que *somos todos diferentes e mesmo assim todos possuimos o direito de usar os sites*.
- Outras razões:
  - Em alguns países é lei.
  - Você não quer excluir clientes/visitantes potenciais de usar seu site.
  - Sites acessíveis tendem a ficar em posições melhores em sites de busca.
  - Criar sites com padrões web facilita bastante torná-los também acessíveis. Facilita a compatibilidade com dispositivos móveis.
  - Técnicas que ajudam pessoas com necessidades especiais beneficiam todos os usuários.

Legalidades da acessibilidade

- Nos estados unidos [1], reino unido e união européia os sites desenvolvidos pelo governo devem ser acessíveis.

Mercados potenciais

- No ano 2000 a rede de supermercados Tesco criou uma versão alternativa de seu site de compras para pessoas com problemas de visão. Isso aumentou as receitas em 13 milhões de libras por ano, receita que não estava disponível porque o site era inacessível.
- Todas as pessoas possuem as mesmas necessidades.

Engenhos de busca

- Engenhos de busca não são pessoas e desenvolvedores geralmente não consideram como vão ser encontrados pelo Google, Bing, etc.
- Por exemplo, imagens que contém texto não são indexadas. Em HTML, o elemento `<img>` possui uma forma de descrever o conteúdo de uma imagem, o atributo `alt`.

Projetando com acessibilidade em mente

- Devemos então projetar nossos sites com acessibilidade em mente, e não acrescentar acessibilidade posteriomente.
- Isso leva mais tempo, não funciona tão bem e resulta em páginas mais mal feitas.

Requisitos de interoperabilidade

- Varia de situação para situação.
- Existem 4 abordagens para acrescentar interoperabilidade em sites:
  - Progressivamente melhorar seu site, testando pelo suporte.
  - Permite os usuários desabilitar melhorias problemáticas.
  - Prover versões alternativas com o mesmo conteúdo ou funcionalidade.
  - Alertar seus clientes quais tecnologias são usadas e prover exemplos de empresas que suportam essas tecnologias.

Requisitos de interoperabilidade

- Mesmo em ambientes de intranet, é importante utilizar padrões abertos:
  - Os navegadores ou sistemas operacionais podem ser modificados.
  - Departamentos diferentes podem usar programas diferentes, necessitando que todos acessem as informações.

Estrutura semântica

- Utilizar uma estrutura semântica é o primeiro passo.
- Ela provê a infraestrutura para a informação na página.
- Quando não se pode visualizar o estilo da página, a estrutura semântica ajuda a indicar várias coisas sobre a página, como sua posição na hierarquia do documento e como os elementos interagem.
- Exemplo: usar elementos `<ul>` e `<li>` para representar listas.

Conteúdo alternativo

- Prover um conteúdo e navegação acessíveis é essencial.
- Texto pode ser lido, aumentado ou diminuído seu tamanho, modificado seu contraste, dentre diversas outras transformações.
- Atributo `longdesc` referencia uma página com uma descrição detalhada do objeto.
- Imagens decorativas devem possuir atributo `alt` vazio.

Web Content Accessibility Guidelines 1.0

- É o padrão de uso mais disseminado para acessibilidade na web [2] desenvolvido pelo W3C [3].
- É um conjunto de 14 diretrizes que tentam encapsular os objetivos necessários para alcançar uma página web acessível.
- Dentro de cada diretriz existem itens a serem checados, que são o objetivo do documento.
- Enquanto as diretrizes explicam os conceitos, os itens são usados para verificar a conformidade.
- Cada item varia de prioridade 1 a 3, para ilustrar sua importância.

Nota	A	AA	AAA
Prioridade	1	1 e 2	1, 2 e 3

Web Content Accessibility Guidelines 2.0

- Menos focado na tecnologia.
- Baseado em quatro princípios de acessibilidade [4]:
  - Percebível: pessoas podem acessar o conteúdo através do meio disponível a ele. Por exemplo, pessoas cegas devem ser capazes de ouvir o conteúdo.
  - Operável: pessoas podem interagir com a aplicação ou conteúdo web
  - Entendível: o conteúdo e a interface com o usuário é entendida pelas pessoas que irão utilizá-la.
  - Robustez: qualquer solução deve estar disponível em diversas plataformas e sistemas. Soluções que a maioria das pessoas não vai usar por causa de hardware/software restrito ou muito caro devem ser evitadas.
- Mais sobre esta aula [5].

Referências

[1] U. S. General Services Administration Federal Government. Section 508, 2015. URL http://www.section508.gov/.

[2] W3C. Web Content Accessibility Guidelines 1.0, 1999. URL http://www.w3.org/TR/WCAG10/.

[3] W3C. Web Accessibility Initiative, 2008. URL http://www.w3.org/WAI/.

[4] W3C. Web Content Accessibility Guidelines 2.0, 2008. URL http://www.w3.org/TR/WCAG20/.

[5] Web Education Community Group. Accessibility basics, 2013. URL http://www.w3.org/community/webed/wiki/Accessibility_basics.

Parte XXV

Testando Acessibilidade

Introdução I

- Teste de acessibilidade na web é um subconjunto do teste de usabilidade onde o usuário em questão possui necessidades especiais que afetam sua forma de navegar na web. O objetivo final, tanto na usabilidade e acessibilidade, é descobrir quão facilmente os usuários usam o site e utilizar essas informações para melhorar os designs e implementações futuras.
- De forma geral, a avaliação de acessibilidade é mais formalizado que a de usabilidade. Existem padrões de acessibilidade na web que podem ser seguidos como a legislação *Section 508* (Estados Unidos) [9] e *Web Content Accessibility Guidelines*, do W3C [12, 13].

Introdução II

- É importante distinguir entre satisfazer um padrão e maximizar a acessibilidade de um site. Idealmente, ambos deveriam ser o mesmo, mas qualquer padrão pode falhar em:
  - atender as necessidades de pessoas de todas as necessidades.
  - balancear as necessidades de pessoas com diferentes deficiências.
  - atender aquelas necessidades com as melhores técnicas.
  - usar linguagem clara para expressar necessidades e técnicas.

Quando testar?

- *"Testar cedo, testar sempre"*. Deixar os testes para o final do processo de desenvolvimento possui dois riscos:
  1. Projetos tendem a ultrapassar o tempo e custo estimados. Testar acaba sendo feito de forma rápida, omitida, ou ignorada graças a tais pressões.
  2. Dá mais trabalho corrigir problemas descobertos tardiamente do que fazer as coisas corretamente de começo.

Entendendo seus requisitos

- Antes de começar a avaliar o projeto, deve-se determinar os requisitos chave, dado o ambiente, audiência, e recursos. Alguns requisitos virão de terceiros, como governo e clientes; outros você poderá escolher.
- Geralmente requisito vem de fontes externas, como:
  - **Governos**: geralmente na forma de legislação contra discriminação, não como obrigação de seguir um determinado padrão.
  - **Políticas do cliente**: empresas contratantes tentam garantir que seus sites estejam em conformidade com um determinado padrão.
  - **Utilidade de marketing**: conformidade com um determinado padrão pode ajudar a vender um projeto.
  - **Políticas internas de acessibilidade**: a própria empresa pode exigir a conformidade com um padrão.

Entendendo seus requisitos
Os detalhes da conformidade

- Alguns padrões possuem mais de um nível de conformidade. Qual deles deve ser atingido?
- WCAG 1.0 possui três níveis de conformidade: A, AA e AAA.
- WCAG 2.0 também possui três níveis, mas as possibilidades de conformidade são mais complexas. Se um recurso faz parte de um processo, o nível de conformidade para todos os recursos do processo é dado pelo recurso de menor nível.
- Afirmações de conformidade devem ser baseadas em tecnologias que suportem acessibilidade. Para isso, ela deve:
  - ter sido demonstrada que trabalha com a tecnologia assistiva do usuário.
  - possuir agentes (navegadores, plugins, etc.) que trabalhem com a tecnologia assistiva e estejam disponíveis aos usuários com necessidades especiais a um custo que não seja mais alto do que aqueles disponíveis a usuários sem necessidades especiais.

Entendendo seus requisitos
Excedendo as expectativas

- Os requisitos externos devem ser tratados como o mínimo aceitável.
- Ao se entregar um relatório, deve-se distinguir entre os requisitos mínimos e os requisitos acrescentados pela equipe de desenvolvimento.
- Seguir um padrão não necessariamente previne as melhores práticas de outro padrão.

- Mesmo se a interface com o usuário não estiver completamente acessível, ele pode identificar o conteúdo de interesse e procurar uma ajuda externa para convertê-lo em um formato que ele possa usar.
- Por exemplo, pessoas com deficiências auditivas podem ter problemas para ouvir um vídeo pela web. Através do endereço do vídeo, ele pode então submetê-lo a um serviço que gera as legendas automaticamente.

Entendendo seus requisitos I
Personas com necessidades especiais

- Criar personagens que agem como arquétipos de tipos particulares de usuários que usarão o site. Para um site de compartilhamento de vídeos poderíamos incluir:
  - rapaz de 23 anos, que gosta de futebol e deseja compartilhar notícias de esporte com colegas.
  - mãe trabalhadora de 34 anos que normalmente não tem tempo para ver vídeos mas sua filha de 3 anos gosta, e ela gostaria de ajudá-la a encontrar vídeos adequados para verem juntas.
- Podemos considerar essas personas e acrescentar necessidades especiais:
  - Problemas de visão
  - Daltonismo
  - Cegueira

Entendendo seus requisitos II
Personas com necessidades especiais

- Surdez
- Problemas de audição
- Cego e surdo
- Epilepsia
- Dislexia

- Se você precisa escolher um padrão de acessibilidade, o mais indicado é o WCAG 2.0, por que:
  - foi projetado baseado nas necessidades humanas básicas que são aplicáveis a tecnologias que não somente HTML e CSS (como Flash).
  - documenta as razões para cada critério de conformidade.
  - sugere técnicas práticas para atingir os critérios de conformidade usando as tecnologias correntes.
  - garante que cada disposição é testável.
  - incorpora pesquisas mais recentes do que as alternativas correntes.
  - foi projetado para ser compatível com os padrões de acessibilidade anteriores.
  - é uma recomendação internacional.

Entendendo seus requisitos
O espírito da lei

- Quando estiver realizando testes, o mais importante é satisfazer o espírito, e não a letra do padrão.
- Por exemplo, o WCAG 1.0 possui um requisito que todo elemento não textual deve conter uma descrição textual equivalente. Obviamente, a qualidade do texto depende do implementador.

```
<img alt="fancy border" src="fancy-border.gif" border="0">
```

- Já no WCAG 2.0, elementos de formatação ou decoração não precisam de um equivalente textual.

Quem deve testar?

- Basicamente dois grupos: usuários e especialistas.
- Teste do especialista é importante pois eles entendem como as tecnologias da web interagem, podendo agir para dirimir dúvidas de diferentes grupos de usuários, e possui a inclinação de aprender ferramentas de teste.
- Teste de usuário é importante pois eles são os reais especialistas em suas habilidades e na tecnologia assistiva. Esse teste pode revelar problemas de usabilidade entre usuários com maior e menor experiência técnica, e entre pessoas que estão familiares com o site em questão e as que não estão.

Teste do especialista

- Existem quatro componentes:
  - Avaliação por uma ferramenta – uma ferramenta procura por problemas de acessibilidade.
  - Emulação – o especialista simula um usuário usando alguma tecnologia assistiva.
  - Inspeção com ferramenta – o avaliador usa uma ferramenta para verificar como as partes do site trabalham em conjunto.
  - Revisão do código – o avaliador analisa o código e recursos de um site procurando por problemas.

Teste do especialista
Verificadores de acessibilidade semi-automatizados

- Após as correções dos problemas mais básicos, é importante usar uma ferramenta verificadora de acessibilidade.
- Para os padrões Section 508 e WCAG 1.0, existe o *Cynthia Says* [3]. No Brasil, temos o *daSilva* [2].
- Para o padrão WCAG 2.0, existe o *IDI Web Accessibility Checker* [1] e o *TAW* [8].
- Essas ferramentas possuem limitações. Em algumas, caso o texto do atributo `alt` estiver vazio, a ferramenta indicará uma advertência.

- Ferramentas podem também verificar a estrutura do documento, quais seus elementos e como eles se relacionam, e não somente descrever como a página é visualmente.
- Ferramentas como o *Web Accessibility Toolbar for Internet Explorer and Opera* [16] e o *ICITA Firefox Accessibility Toolbar* [4].

Teste do especialista
Emulação e usando tecnologia assistiva do usuário

- Emulação pode envolver:
  - Usar um palito para pressionar teclas durante a navegação para testar a acessibilidade do teclado.
  - Visualizar a página com o simulador *Vischeck* [11], que tenta apresentar a página, incluindo as imagens, para pessoas com diferentes níveis de deficiência visual.
  - Desligar o monitor enquanto usando um leitor de tela em conjunto com um navegador.
- Usar tecnologia assistiva não é uma tarefa fácil, pois requer um nível de imersão e treinamento. Existe um sério risco de concepções incorretas.
- Ler uma página que se pode ver ou que se conhece é diferente de explorar uma página completamente desconhecida.

Teste do especialista
Inspeção detalhada

- Após a correção dos erros apontados pelos verificadores, passamos aos testes manuais, sondagem, e revisão do projeto.
- WCAG 2.0 divide seus critérios de melhores práticas em quatro princípios. Conteúdo e funcionalidade devem ser:
  - Percebíveis (por exemplo, imagens devem possuir equivalentes textuais).
  - Operáveis (por exemplo, deve ser possível interagir com o site sem um mouse e navega-lo com um leitor de tela).
  - Entendível (por exemplo, cópia não deve ser mais complicada do que o necessário e o site deve operar de forma previsível).
  - Robusto (por exemplo, web sites devem trabalhar de forma interoperável com diferentes agentes de usuário e navegação deve ser consistente).

Percebível

- Um grupo de problemas envolve a utilização de conteúdo alternativo de vários tipos. Deve-se prestar atenção especial aos elementos `<img>` e `<input>`. Normalmente deve-se usar o atributo `alt` vazio para elementos puramente de decoração. Entretanto, nos casos de:
  - Imagens que são o único conteúdo de links
  - Botões de formulários
- Se estes elementos possuírem o atributo `alt` vazio, leitores de tela geralmente irão tratar como se o atributo estivesse faltando, e tentar prover um (por exemplo, lendo a URL da imagem).

Percebível

- Outro problema está relacionado a estilizar a página. Existem três áreas a investigar:
  - A apresentação da página está sensatamente acessível? Por exemplo, existe contraste de cor suficiente? O texto está confortavelmente largo? Ferramentas como o *Juicy Studio CSS Analyser* [6] analisa as cores de primeiro plano com as de fundo para medir a legibilidade.
  - Verificar como a apresentação se comporta com mudança no tamanho das fontes e cores diferentes. Se o CSS muda as cores, deve faze-lo tanto com as primeiro plano como com as de fundo, de forma a evitar em texto ilegível ou invisível.
  - Se o CSS for desabilitado, o conteúdo ainda continua disponível de forma legível?

Operável

- Evitar conteúdos piscantes, que podem afetar pessoas com epilepsia. A ferramenta *Trace Center Photosensitive Epilepsy Analysis Tool* [10] testa se o site possui conteúdo que possa gerar perigo aos usuários.
- Verificar se o conteúdo e funcionalidades estão acessível através de dispositivos distintos:
  - Tente usar o site apenas com o teclado. O foco está sempre claramente indicado? Toda a funcionalidade pode ser acessada pelo teclado?
  - Tente usar o site com um dispositivo *touchscreen*.
  - Tente navegar pelo site com comandos de voz.
- Leitores de tela e tecnologias assistivas usam a estrutura do documento (X)HTML para associar os elementos e permitir a navegação do conteúdo. Usar os elementos `<label>` e `<legend>` para associar campos do formulário, `<th>`, `header`, `scope`, e `axis`, para associar uma célula da tabela com seu cabeçalho, etc.

Compreensível

- Garantir que o texto está compreensível é ainda mais subjetivo do que testar a legibilidade. Pode-se usar uma ferramenta como *Juicy Studio's Readability Test* [7] que dá uma ideia de quão simples seu site é.
- Outros critérios são objetivos, como se o conteúdo possui a indicação do idioma utilizado, de forma a leitores de tela lerem o texto com o sotaque correto.

Robusto

- Envolve checar se as tecnologias estão sendo usadas corretamente. De forma básica, testar com o *W3C HTML Validator* [15] com advertências habilitadas, *W3C CSS Validator* [14] e *JSLint JavaScript* [5].
- Verificar se JavaScript usa detecção de características ao invés de detecção do navegador.
- O problema mais comum provavelmente é JavaScript obstrusivo, como âncoras e botões que estão versão sem scripts mas dependem de JavaScript para funcionar.

Teste do usuário

- Nenhuma quantidade de inspeção do desenvolvedor e emulação pode substituir a interação de um usuário com o site.
- Deve-se recrutar testadores com necessidades especiais.
- O ambiente de teste deve estar acessível (por exemplo, o material de teste escrito disponível em braille).
- Deve-se replicar o ambiente do usuário em laboratório (teste alfa) ou testar na casa do usuário (teste beta), ou ainda testar de forma remota.
- Que tecnologias usar? Existe uma divisão entre pessoas com conhecimento de uma determinada tecnologia assistiva e outros que não tem.

Teste do usuário

- É bastante instrutivo observar como os usuários exploram o site. Como em qualquer teste de usuário:
  - Tente indicar tarefas específicas que eles devem realizar.
  - Pergunte o que eles pensam e ouça o que eles dizem.
  - Preste atenção ao que eles fazem, que pode diferir do que eles falam.

Informando o idioma do documento

- Colocado no elemento <html>, indica o idioma do documento.
- Ajuda leitores de tela e engenhos de busca.
- Pode-se também indicar o idioma de subseções de seu documento.
- Mais sobre esta aula [17].

```
<html lang="en-GB">
  ...
</html>
```

Referências I

[1] AChecker. IDI Web Accessibility Checker, 2011. URL http://achecker.ca/checker/index.php.

[2] DaSilva. Avaliador de Acessibilidade para Websites, 2015. URL http://www.dasilva.org.br/.

[3] HiSoftware. Cynthia Says, 2015. URL http://www.cynthiasays.com/.

[4] Illinois Center for Information Technology and Web Accessibility. Firefox Accessibility Extension, 2008. URL http://firefox.cita.uiuc.edu/.

[5] JSLint. The JavaScript Code Quality Tool, 2002. URL http://www.jslint.com/.

[6] Juicy Studio. CSS Analyzer, 2015. URL http://juicystudio.com/services/csstest.php.

Referências II

[7] Juicy Studio. Readability Test, 2015. URL http://juicystudio.com/services/readability.php.

[8] TAW. Servicios de Accesibilidad y Movilidad Web, 2015. URL http://www.tawdis.net/.

[9] U. S. General Services Administration Federal Government. Section 508, 2015. URL http://www.section508.gov/.

[10] University of Wisconsin. Photosensitive Epilepsy Analysis Tool, 2010. URL http://trace.wisc.edu/peat/.

[11] Vischeck. Vischeck, 2008. URL http://www.vischeck.com/.

[12] W3C. Web Content Accessibility Guidelines 1.0, 1999. URL http://www.w3.org/TR/WCAG10/.

[13] W3C. Web Content Accessibility Guidelines 2.0, 2008. URL http://www.w3.org/TR/WCAG20/.

Referências III

[14] W3C. CSS Validation Service, 2015. URL http://jigsaw.w3.org/css-validator/.

[15] W3C. The W3C MarkUp Validation Service, 2015. URL http://validator.w3.org/.

[16] WAT-C. Web Accessibility Tools Consortium, 2005. URL http://www.wat-c.org/tools/.

[17] Web Education Community Group. Accessibility testing, 2013. URL http://www.w3.org/community/webed/wiki/Accessibility_testing.

Parte XXVI

Introdução a Javascript?

O que é Javascript e como executá-lo?

- Javascript é uma linguagem textual que não precisa de conversão para ser executado.
- Ela é interpretada por um parser do navegador.
- Tem por objetivo acrescentar uma camada de comportamento, ou seja, possibilitar e facilitar a interação da página com o usuário.
- Podemos executá-la dentro de um elemento `<script>` em qualquer ponto do documento HTML ou colocá-lo em um arquivo em separado (com a extensão `.js` e referenciá-lo usando o elemento `<script>` com o atributo `src`.

Os problemas de Javascript

- Não a linguagem mas o ambiente em que é executado. Não se sabe qual computador irá executar a página, o quão ocupado ele está, ou se outro código Javascript em outra aba está influenciando na performance no navegador.
- Javascript pode estar desabilitado no cliente por questões de segurança ou para melhorar a experiência do usuário.

Incluindo o Javascript no corpo do documento

- A inclusão mais básica de Javascript diretamente no corpo do documento é assim

```
<script type="text/javascript">
  var x = 3;
  alert('hello there, I am JavaScript - x is ' + x);
</script>
```

- No passado, existia a necessidade de comentar o código Javascript para evitar que os navegadores mostrassem o código como HTML. Isso não é mais necessário. Se estivermos usando XHTML Strict, devemos encapsular o Javascript em um bloco CDATA para ele validar.

```
<script type="text/javascript">
/* <![CDATA[ */
  var x = 3;
  alert('hello there, I am JavaScript - x is ' + x);
/* ]]> */
</script>
```

Referenciando um arquivo Javascript externo I

- Para referenciarmos um arquivo externo, só precisamos acrescentar o atributo `src`:

```
<script type="text/javascript" src="myscript.js"></script>
```

- O exemplo a seguir carregará e executará o código `myscript.js`, mas não executará o comando interno ao elemento.

```
<script type="text/javascript" src="myscript.js">
  alert('I am pointless as I won\'t be executed');
</script>
```

- Benefícios de manter o código externo:
  - Pode-se aplicar o mesmo código Javascript a várias páginas mantendo a manutenção fácil.
  - Javascript pode sofrer cache dos navegadores.
  - Pode-se facilmente encontrá-lo e modificá-lo se necessário, evitando procurar em documentos HTML longos.

Referenciando um arquivo Javascript externo II

- Corrigir problemas se torna mais fácil com a utilização de ferramentas de depuração ou console de erros. Elas dirão qual arquivo contém o erro além de reportar a linha.

- Pode-se acrescentar quantos arquivos externos quanto necessários.

Sintaxe I

- Muito semelhante à C e Java.
- Tipagem fraca. Variáveis recebem o tipo do valor atribuído a elas.
- Pode-se utilizar a palavra reservada `var` para indicar a definição de uma variável.

```
var x = 1; // number
y = 'a'; // string
```

- Comando `alert` abre uma janela com o valor passado; `console.log` informa no console de erros no navegador (F12).

```
alert(x);
console.log(y);
```

- Comando `typeof` informa o tipo de uma variável.

```
alert(typeof x);
console.log(typeof(y));
```

Sintaxe II

- Variável sem valor explícito possui valor e tipo **undefined**.

```
var z;
console.log(z);
console.log(typeof z);
```

- Os operadores são basicamente os mesmos de C e Java.
- O operador == verifica se ambos os valores são os mesmos, tentando conversão de tipos. === compara valores de um mesmo tipo.

```
alert(10 == '10'); // true
alert(10 === '10'); // false
```

- Mais sobre esta aula em [1–4].

Referências I

[1] Web Education Community Group. Programming - the real basics, 2011. URL http://www.w3.org/community/webed/wiki/Programming_-_the_real_basics.

[2] Web Education Community Group. What can you do with JavaScript, 2011. URL http://www.w3.org/community/webed/wiki/What_can_you_do_with_JavaScript.

[3] Web Education Community Group. Your first look at JavaScript, 2011. URL http://www.w3.org/community/webed/wiki/Your_first_look_at_JavaScript.

[4] Web Education Community Group. A Short History of JavaScript, 2012. URL http://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript.

Parte XXVII

Funções em Javascript

Sintaxe

1. A declaração de uma função sempre começa com a palavra chave `function`.
2. Em seguida vem o nome da função.
3. Após o nome segue a lista de argumentos, que vem entre parênteses.
4. Finalmente, o corpo da função vem entre chaves.

```
function setElementBackground() {
  var red = Math.floor(Math.random() * 256);
  var green = Math.floor(Math.random() * 256);
  var blue = Math.floor(Math.random() * 256);

  var obj = document.getElementById('element_to_change');
  if ( obj ) {
    obj.style.background = 'rgb(' + red + ',' + green + ',' + blue + ')
        ';
  }
}
```

Sintaxe I

- Para usar a função, basta chamá-la pelo nome no código

  `setElementBackground();`

- Para passar argumentos para a função, basta informar o nome da variável.

```
function setElementBackground( elementID ) {
  var red = Math.floor(Math.random() * 256);
  var green = Math.floor(Math.random() * 256);
  var blue = Math.floor(Math.random() * 256);

  var obj = document.getElementById( elementID );
  if ( obj ) {
    obj.style.background = 'rgb(' + red + ',' + green + ',' + blue + ')
        ';
  }
}
```

Sintaxe II

- Para chamar a função com argumentos:

```
setElementBackground( 'element_to_change');
```

- Se acidentalmente chamar a função sem argumentos, ela receberá o valor **undefined**. Podemos então testá-la dentro da função:

```
if ( elementID == undefined) {
  // This will evaluate to `true` if the `elementID`
  // variable wasn't provided by the caller.
  // You can then write some code inside this
  // if statement to stop the code from erroring.
}
```

- Os nomes das variáveis na lista de argumentos não tem relação com o nome das variáveis passadas para a função.

```
var elementID = "No change!";
setElementBackground( 'element_to_change' );
alert( elementID ); // Alerts "No change!";
```

Sintaxe III

- Para retornar valores, usamos a palavra reservada **return**.

```
function setElementBackground( elementID ) {
  var red = Math.floor(Math.random() * 256);
  var green = Math.floor(Math.random() * 256);
  var blue = Math.floor(Math.random() * 256);

  var obj = document.getElementById( elementID );
  if ( obj ) {
    obj.style.background = 'rgb(' + red + ',' + green + ',' + blue + ')
        ';
  }

  return [ red, green, blue ];
}

var my_result = setElementBackground('element_to_change');
```

Parâmetros de uma função

1. Funções possuem um objeto criado automaticamente chamado `arguments`, que é um vetor com os argumentos usados na função.

```
x = findMax(1, 123, 500, 115, 44, 88);

function findMax() {
    var i, max = 0;
    for (i = 0; i < arguments.length; i++) {
        if (arguments[i] > max) {
            max = arguments[i];
        }
    }
    return max;
}
```

Escopo

1. Variáveis declaradas com `var` possuem escopo local; sem ela, possuem escopo global.

```
// x não pode ser usado aqui
var y = 0;
// y pode ser usado aqui
// z pode ser usado aqui (após execução da função)
function myFunction() {
    var x = 1;
    z = 2;
    // x pode ser usado aqui
    // y pode ser usado aqui
    // z pode ser usado aqui
}
```

Função de alta ordem

1. Em Javascript, funções são um tipo da linguagem. Sendo assim, podem ser passadas por parâmetro ou retornadas por outras funções. Isso permite encapsular o comportamento, parametrizando-o.
2. Exemplos em vetores:
   1. `forEach`: Aplica uma função a todos os elementos.
   2. `filter`: cria um novo vetor com os elementos que satisfazem a um requisito.
   3. `map`: aplica uma função a todos os elementos e constrói um novo vetor com os valores retornados pela função.
   4. `reduce`: calcular um valor a partir dos elementos do vetor.
   5. `sort`: ordena um vetor a partir de uma função de ordenação.
3. Mais sobre esta aula em [1, 2].

Referências I

[1] Web Education Community Group. JavaScript functions, 2011. URL http://www.w3.org/community/webed/wiki/JavaScript_functions.

[2] Web Education Community Group. JavaScript best practices, 2011. URL http://www.w3.org/community/webed/wiki/JavaScript_best_practices.

Parte XXVIII

Objetos em Javascript

Objetos I

- Permite organizar e reutilizar código agrupando atividades individuais em blocos lógicos que podem ser chamados sempre que necessário.
- Tem a capacidade de melhorar a representação dos dados e processos.
- Tentemos representar um triângulo. Sabemos que ele possui três lados e podemos calcular sua área.

```
// This is a triangle.
var sideA = 3;
var sideB = 4;
var sideC = 5;

function getArea( a, b, c ) {
  // Return the area of a triangle using Heron's formula

  var semiperimeter   =   (a + b + c) / 2;
  var calculation     =   semiperimeter * (semiperimeter - a) * (
      semiperimeter - b) * (semiperimeter - c);
  return Math.sqrt( calculation );
}
```

Objetos II

```
alert( getArea( sideA, sideB, sideC ) );
```

- Devemos passar todas as informações sobre o triângulo para a função. As atividades estão desacopladas dos dados. E se quisermos representar um retângulo?
- Para criar um objeto que represente um triângulo, podemos criá-lo explicitamente da seguinte forma:

```
var triangle = new Object();
triangle.sideA = 3;
triangle.sideB = 4;
triangle.sideC = 5;
```

Objetos III

- Quando Javascript avalia o operador ponto, se a propriedade não existir, ela é criada. Se tentar ler uma propriedade que não exista, Javascript retorna **undefined**. Adicionar métodos é semelhante, através de uma função anônima. Ela é uma função que não possui nome mas que é armazenada em uma variável como qualquer outro valor.

```
triangle.getArea = function ( a, b, c ) {
  // Return the area of a triangle using Heron's formula

  var semiperimeter = (a + b + c) / 2;
  var calculation   = semiperimeter * (semiperimeter - a) * (
      semiperimeter - b) * (semiperimeter - c);
  return Math.sqrt( calculation );
}; // Note the semi-colon here; it's mandatory.
```

Referência própria I

- Um dos objetivos da criação do objeto triângulo foi estabelecer uma ligação entre os dados do triângulo e as ações que posso realizar neles. Isso ainda não foi atingido. O método `triangle.getArea` ainda requer que os tamanhos dos lados sejam passados por parâmetro. Podemos fazer este método acessar os dados diretamente. Para isso usaremos a palavra chave `this`.

```
triangle.getArea = function () {
  // Return the area of a triangle using Heron's formula

  var semiperimeter = (this.sideA + this.sideB + this.sideC) / 2;
  var calculation   = semiperimeter * (semiperimeter - this.sideA) * (
      semiperimeter - this.sideB) * (semiperimeter - this.sideC);

  return Math.sqrt( calculation );
}; // Note the semi-colon here, it's mandatory.
```

Objetos como vetores associativos I

- O operador ponto não é a única forma de acessar as propriedades e métodos de um objeto; eles também podem ser acessados de forma eficiente usando a **notação subscrita**, semelhante a vetores. Podemos então tratar um objeto como um vetor associativo que mapeia uma string em um valor da mesma forma que vetores mapeiam um número (índice) para um valor. Outra forma de representar o triângulo seria:

```
var triangle = new Object();
triangle['sideA']    =    3;
triangle['sideB']    =    4;
triangle['sideC']    =    5;
triangle['getArea'] =    function ( a, b, c ) {
  // Return the area of a triangle using Heron's formula

  var semiperimeter    =    (a + b + c) / 2;
  var calculation      =    semiperimeter * (semiperimeter - a) * (
      semiperimeter - b) * (semiperimeter - c);
```

Objetos como vetores associativos II

```
  return Math.sqrt( calculation );

}; // Note the semi-colon here, it's mandatory.
```

- Esta notação permite que as propriedades não estejam descritas diretamente no código. Pode-se usar variáveis para especificar os nomes das propriedades.
- Por exemplo, podemos contruir uma função que compare dois objetos para verificar se eles compartilham uma propriedade comum:

```
function isPropertyShared( objectA, objectB, propertyName ) {
  if(typeof objectA[ propertyName ] !== undefined && typeof objectB[
      propertyName ] !== undefined) {
    alert("Both objects have a property named " + propertyName + "!");
  }
}
```

Objeto literal I

- Da mesma forma que podemos criar uma string diretamente: `var s = "Hello World!";`, podemos criar objetos diretamente. Notação chamada **literal**.

```
var triangle = {
  sideA:   3,
  sideB:   4,
  sideC:   5,
  getArea: function () {
    // Return the area of a triangle using Heron's formula

    var semiperimeter = (this.a + this.b + this.c) / 2;
    var calculation   = semiperimeter * (semiperimeter - this.a) * (
        semiperimeter - this.b) * (semiperimeter - this.c);
    return Math.sqrt(calculation);

  }
};
```

Objeto literal II

- A sintaxe é clara: objeto literal usa chaves para demarcar o começo e fim de um objeto, que contém um número arbitrário de pares "nomePropriedade: valorPropriedade" separados por vírgulas.

Vetores

- Funciona de forma similar a Java. Podemos criar um vetor de duas formas distintas.
- Diretamente: `var cars = new Array("Saab", "Volvo", "BMW");`
  - Construtor com um único inteiro: `var cars = new Array(40);` Cria um vetor com 40 elementos indefinidos.
- Notação chamada **literal**: `var cars = ["Saab", "Volvo", "BMW"];`
  - Melhor opção.
- Mais sobre esta aula em [1].

Referências I

[1] Web Education Community Group. Objects in JavaScript, 2011. URL http://www.w3.org/community/webed/wiki/Objects_in_JavaScript.

Parte XXIX

Percorrendo a árvore HTML

Introdução

- É difícil encontrar um exemplo útil em Javascript que não interaja com o documento HTML.
- Geralmente o código precisa ler valores na página, processá-los e gerar a saída na forma de mudanças visíveis ou mensagens.
- DOM (**Document Object Model**) provê mecanismos para *inspecionar* e *manipular* as camadas semântica e de apresentação.

Plantando as sementes

- DOM é o modelo do documento HTML que é criado pelo navegador quando ele carrega a página. Javascript tem acesso a todas as informações neste modelo.
- Pode-se encontrar um elemento ou grupos de elementos e removê-los, acrescentá-los, e modificá-los de acordo com valores definidos pelo usuário; informações de apresentação também poderão ser manipuladas; validar dados de formulário; dentre outras.
- Criar documentos HTML e CSS bem estruturados formam a base na qual o modelo de Javascript se desenvolverá.

Árvores

- DOM representa a página HTML no formato de árvore, da mesma forma que uma árvore genealógica.
- Cada elemento na página é representado em DOM como um nó, com ramos ligando a elementos que ele contém (seus **filhos**), e ao elemento que o contém (seu **pai**).

Nós

- Cada nó na árvore DOM é um objeto representando um único elemento na página. Nós conhecem seus relacionamentos para outros nós vizinhos, e contém informações sobre si mesmos.
- Essas informações estão expostas via propriedades dos nós. Especificamente, as propriedades `parentNode` e `childNodes`.
- Como cada elemento só possui um pai, `parentNode` retorna o nó pai.
- Nós podem ter muitos filhos, então `childNodes` retorna um vetor, com os filhos na mesma ordem que aparecem no documento.
- A raiz do documento está acessível através do objeto `document`. Ele não possui pai e possui somente um filho, o elemento `html`.

Propriedades de um nó I

- Para informar o nome do nó, utilizamos a propriedade `nodeName`
- Para saber o tipo de nó (elemento, atributo, texto, etc.), utilizamos a propriedade `nodeType`
- Para saber o valor de um nó (que varia dependendo do tipo), utilizamos a propriedade `nodeValue`

Acesso direto I

- Podemos utilizar métodos para acessar diretamente os elementos desejados.
- Caso o objeto possua o atributo `id`, que é único na página, podemos acessá-lo usando o método `getElementById` dentro do objeto `document`. Se o `id` não existir, o método retorna `null`.
- Outro método muito útil é o `getElementsByTagName`, que retorna uma coleção com todos os elementos na página de um determinado tipo. Podemos obter todos os parágrafos da página assim:

```
var allParagraphs = document.getElementsByTagName('p');
```

- Caso os elementos possuam um mesmo valor para o atributo `class`, pode-se utilizar o método `getElementsByClassName`.
- O método mais genérico de busca de elementos é o `querySelectorAll`, onde passamos um seletor CSS como pesquisa. `querySelector` retorna apenas o primeiro resultado.

Acesso direto II

- Pode-se também obter os elementos dentro de uma determinada seção do texto. Por exemplo, todos os elementos `em` dentro de um parágrafo específico:
- Mais sobre esta aula em [1].

```
document.getElementById('excitingText').getElementsByTagName('em')
```

Referências I

[1] Web Education Community Group. Traversing the DOM, 2011. URL http://www.w3.org/community/webed/wiki/Traversing_the_DOM.

Parte XXX

Gerenciando eventos em Javascript

O que são eventos?

- Eventos ocorrem quando alguma forma de interação acontece na página. Pode ser um usuário clicando em um botão, movendo o mouse sobre um elemento ou pressionando algumas teclas no teclado. Um evento também pode ser algo que aconteceu com o navegador, como a página terminando de carregar, o usuário usando o scroll ou redimensionando a janela.
- Através de Javascript podemos detectar quando certos eventos ocorrem e executar código em resposta a esses eventos.

Como os eventos funcionam

- Quando eventos acontecem em um elemento HTML na página, ele verifica se algum evento está associado a ele. Se sim, ele é chamado, enviando informações sobre o evento.
- Existem dois tipos de ordem que os eventos podem executar: *event capturing* e *event bubbling*.
- *Event capturing* começa com o elemento DOM mais externo e desce na árvore até o elemento que gerou o evento.
- *Event bubbling* é o oposto: começa no alvo do evento subindo até encontrar o elemento HTML.

A evolução dos eventos

- No começo a utilização de Javascript, os eventos eram acrescentados diretamente ao elemento HTML:

```
<a href="http://www.opera.com/" onclick="alert('Hello')">Say hello</a>
```

- O problema é que resultava em eventos dispersos pelo código, sem controle central e não utilizando funcionalidade de cache dos navegadores para Javascripts externos.
- O próximo passo na evolução foi aplicar eventos dentro de blocos Javascript:

```
<script type="text/javascript">
document.getElementById("my-link").onclick = waveToAudience;
function waveToAudience() {
  alert("Waving like I've never waved before!");
}
</script>

<a id="my-link" href="http://www.opera.com/">My link</a>
```

Eventos DOM Nível 2

- W3C oferece uma forma mais detalhada para controlar eventos. Abaixo temos um exemplo.
- O primeiro parâmetro do método `addEventListener` é o nome do evento (sem o prefixo "on"). O segundo parâmetro é a função a ser executada na ocorrência do evento. O terceiro parâmetro indica se será usado *event capturing* (`true`) ou *event bubbling* (`false`).

```
document.getElementById("my-link").addEventListener("click", myFunction
    , false);
```

- Para remover um evento, usamos o método `removeEventListener`.

Eventos e acessibilidade

- Em geral, nunca aplique eventos a elementos HTML que não possuem comportamento padrão para aquele evento. Você deveria apenas aplicar eventos `onclick` a elementos como `a`, que possuem um comportamento padrão (navegar para a página do link, ou submeter o formulário).

Aplicando eventos a elementos específicos

- Vamos aplicar um evento a todos os links quando forem clicados.

```
window.addEventListener("load", function () {
  var links = document.getElementsByTagName("a");
    for (var i=0; i<links.length; i++) {
      links[i].addEventListener("click", function () {
        alert("NOPE! I won't take you there!");

        // This line's support added through the addEvent function. See
            below.
      evt.preventDefault();
    }, false);
  }
}, false);
```

Referências ao objeto evento

- Podemos tomar ações dependendo de certas propriedades do evento. Por exemplo, lidando com `onkeypress`, queremos executar o evento somente se o usuário pressionar a tecla Enter.
- O W3C recomenda passar um objeto com as informações sobre o evento por parâmetro à função registrada.
- Vejamos como obter uma referência para o objeto e para o elemento que gerou o evento:

```
document.getElementById("check-it-out").addEventListener("click",
    eventCheck, false);
function eventCheck (event) {
  alert(event.target);
}
```

Verificando uma propriedade específica do evento

- O exemplo abaixo executa blocos de código distintos dependendo da tecla que foi pressionada:

```
document.getElementById("user-name").addEventListener("keyup", whatKey,
     false);
function whatKey (event) {
  var keyCode = event.keyCode;
  if (keyCode === 13) {
    // The Enter key was pressed
    // Code to validate the form and then submit it
  }
  else if (keyCode === 9) {
    // The Tab key was pressed
    // Code to, perhaps, clear the field
  }
}
```

Prevenindo o comportamento padrão dos eventos

- Existem situação que se deseja mudar o comportamento padrão de um evento. Por exemplo, pode-se desejar prevenir o usuário de submeter um formulário se alguns campos não foram preenchidos.

```
document.getElementById("stop-default").addEventListener("click",
    stopDefaultBehavior, false);

function stopDefaultBehavior (event) {
    event.preventDefault();
}
```

Parar *event bubbling* I

- Considere o seguinte código:

```
<ul>
  <li><a href="http://www.opera.com/">Opera</a></li>
  <li><a href="http://www.firefox.com/">Firefox</a></li>
</ul>
```

- Suponha que tenhamos acrescentado eventos `onclick` para todos os elementos `a`, `li` e `ul`. O gerenciador de eventos chamará primeiro o evento do link, depois dos itens da lista, e depois da lista. Caso o link seja clicado, não queremos chamar os eventos dos itens de lista nem da lista.

Parar *event bubbling* II

```
onload = function () {
    var elems = document.querySelectorAll("ul, li");
    for(var i = 0, max = elems.length; i < max; i++) {
        elems[i].addEventListener("click", noClique, false);
    }
    elems = document.getElementsByTagName("a");
    for(var i = 0, max = elems.length; i < max; i++) {
        elems[i].addEventListener("click", cancelEventBubbling,
            false);
    }
    function cancelEventBubbling(event) {
        alert(this);
        event.stopPropagation();
        event.preventDefault();
    }

    function noClique(event) {
        alert(this);
    }
};
```

- Mais sobre esta aula em [1, 2].

Referências I

[1] Web Education Community Group. Handling events with JavaScript, 2011. URL http://www.w3.org/community/webed/wiki/Handling_events_with_JavaScript.

[2] Web Education Community Group. The principles of unobtrusive JavaScript, 2012. URL http://www.w3.org/community/webed/wiki/The_principles_of_unobtrusive_JavaScript.

Parte XXXI

Criando e Modificando HTML

Criando elementos I

- Podemos criar elementos na página usando o método `createElement`.
- Para criar nós de texto podemos utilizar `createTextNode` ou a propriedade `textContent`.
- HTML:

```
<body>
    <div id="a">
        <p id="b">Parágrafo.</p>
    </div>
</body>
```

- Javascript:

```
var elem = document.createElement("p");
elem.appendChild(document.createTextNode("Novo parágrafo."));
elem.textContent = "Novo parágrafo.";
```

Criando elementos II

- Após, devemos informar em que lugar da árvore HTML devemos inserir o elemento recém criado. Podemos inserir como último elemento de um elemento com `appendChild`, inserir antes de um elemento com `insertBefore` ou substituir um nó filho com `replaceChild`

```
// Novo paragrafo após antigo
var x = document.getElementById("a");
x.appendChild(elem);

// Novo paragrafo antes do antigo
var y = document.getElementById("b");
x.insertBefore(elem,y);

// Novo paragrafo substituindo antigo
x.replaceChild(elem, y);
```

Criando elementos III

- Também existe a propriedade `innerHTML` que permite acrescentar a um nó tanto texto quando outros elementos diretamente:

```
elem.innerHTML = "Novo <em>parágrafo</em>.";
```

- Estudar outros métodos em [1]; mais sobre esta aula em [2].

Referências

[1] Peter-Paul Koch. Quirks Mode, 2015. URL http://www.quirksmode.org/compatibility.html.

[2] Web Education Community Group. Creating and modifying HTML, 2011. URL http://www.w3.org/community/webed/wiki/Creating_and_modifying_HTML.

Parte XXXII

Estilo Dinâmico – Manipulando CSS com Javascript

Criando um elemento de estilo

- Podemos também acrescentar novos estilos à página usando a função `document.createElement` para criar um novo elemento `style`. Isso é útil para oferecer aos usuários da página formas de modificar o estilo do site dinamicamente. Vejamos um exemplo:

```
var sheet = document.createElement('style');
sheet.innerHTML = "div {border: 2px solid black; background-color: blue
    ;}";
document.head.appendChild(sheet);
```

Acessando estilos

- O navegador provê uma interface para interagir com folhas de estilo usando `document.styleSheets`. Ela retornará uma lista com todos os estilos aplicados à página, incluindo estilos externos e internos.
- Cada elemento desta coleção é do tipo `CSSStyleSheet`. Ele permite obter informações sobre os estilos da página, como se estão desabilitados, sua localização, e a lista de regras CSS que ela contém.

```
var estilos = document.styleSheets;
var stylesheet = estilos[0];
```

Adicionando e removendo regras

- Podemos manipular as regras CSS dentro de cada folha de estilo.
- O objeto `CSSStyleSheet` possui duas funções para nos ajudar: **`insertRule`**`(rule, index)`, onde `rule` contém a regra e `index` especifica onde a regra será incluída na lista de regras das folhas de estilo.

```
stylesheet.insertRule(".have-border { border: 1px solid black;}", 0);
```

- A outra função é **`deleteRule`**, que recebe o índice da regra a ser removida.

```
stylesheet.deleteRule(0);
```

Lendo as regras

- Podemos também acessar as regras CSS dentro de uma folha de estilo.
- O objeto `CSSStyleSheet` possui o método `cssRules`, que retorna uma `CSSRuleList`.

```
document.styleSheets[0].cssRules[0]
```

- Cada elemento desta coleção é uma `CSSRule`.
- Nele, podemos acessar a propriedade `cssText`, que trará a regra CSS.

Modificando o estilo de elementos I

- Vimos como editar estilos ligados à página e criar e modificar regras CSS neles. E se desejarmos mudar um elemento específico com DOM?
- DOM nos dá acesso ao objeto `style`. Diferentemente de outras propriedades que retornam strings, como `element.href` ou `element.id`, `element.style` retorna um objeto.
- Este objeto possui atributos que correspondem às diferentes propriedades CSS. Por exemplo:

```
function colorElementRed(id) {
  var el = document.getElementById(id);
  el.style.color = "red";
}
```

Modificando o estilo de elementos II

- Poderíamos usar `setAttribute(key, value)`, como em `element.setAttribute('style', 'color: red');`, mas ele apaga os estilos anteriores.
- As propriedades que contém hífens seguem a notação **CamelCase**. Por exemplo:

```
function changeElement(id) {
  var el = document.getElementById(id);
  el.style.color = "red";
  el.style.fontSize = "15px";
  el.style.backgroundColor = "#FFFFFF";
}
```

- Outra forma de mudar o estilo de um elemento é modificar seu atributo `class` usando a propriedade `element.className`.
- Mais sobre esta aula em [1–3].

Referências

[1] Web Education Community Group. Dynamic style - manipulating CSS with JavaScript, 2011. URL http://www.w3.org/community/webed/wiki/Dynamic_style_-_manipulating_CSS_with_JavaScript.

[2] Web Education Community Group. JavaScript animation, 2011. URL http://www.w3.org/community/webed/wiki/JavaScript_animation.

[3] Web Education Community Group. Graceful degredation versus progressive enhancement, 2011. URL http://www.w3.org/community/webed/wiki/Graceful_degredation_versus_progressive_enhancement.