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

12 de agosto de 2020

Parte I

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

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.

Parte II

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

Apresentação dos elementos

- Para informar como um elemento será apresentado, utilizamos a propriedade `display`.
- Os valores mais comuns são `block`, `inline`, `inline-block`, `none`, `flex` e `grid`.
- De forma geral, todos os elementos com associação a conteúdo são dos tipos bloco ou linha.
- Com eles, podemos alterar o comportamento padrão dos elementos.

Apresentação dos elementos

block

- Esse valor informa que o elemento ocupará qualquer largura disponível, independente do seu conteúdo, e sempre começará em uma nova linha.
- Elementos de nível de bloco são geralmente utilizados para grandes pedaços de conteúdo, como cabeçalhos e elementos estruturais.
- Exemplos: `<h1>` a `<h6>`, `<p>`, `<div>`

Apresentação dos elementos

inline

- Esse valor informa que o elemento ocupará apenas a largura que o seu conteúdo necessita e ficará na mesma linha, um após o outro.
- Elementos de nível de linha são geralmente utilizados para pequenos pedaços de conteúdo, como pequenas palavras selecionadas para negrito e itálico.
- Não pode ser usada em conjunto com **`width`** e **`height`**.
- Exemplos: `<strong>`, `<em>`, `<code>`, `<span>`

Apresentação dos elementos

inline-block

- Esse valor informa que o elemento se comportará como um elemento de bloco, aceitando todas as propriedades do modelo de caixa de CSS.
- O elemento será apresentado em linha com os outros elementos, e não começará em uma nova linha por padrão.

Apresentação dos elementos

none

- Esse valor informa que o elemento ficará completamente escondido e renderizará a página como se o elemento não existisse.
- Quaisquer subelementos deste elemento também ficarão escondidos.

Apresentação dos elementos

flex

- Permite informar como elementos aparecerão um ao lado do outro, podendo ocupar mais de uma linha.
- Deve-se informar um elemento pai com o valor `flex` e todos os filhos (descendentes diretos) automaticamente se tornam itens flexíveis.

Apresentação dos elementos

grid

- Oferece um sistema de layout em formato de tabela, com linhas e colunas.
- Deve-se informar um elemento pai com o valor `grid` e todos os filhos (descendentes diretos) automaticamente se tornam itens do layout.