中山大學
SUN YAT-SEN UNIVERSITY

Lecture 03. HTML & CSS II

Modern Web Programming
(http://my.ss.sysu.edu.cn/wiki/display/WEB/ **supported by Deep Focus**)

School of Data and Computer Science, Sun Yat-sen University

Part I

Outline

- **More CSS**
- Styling Page Sections
- Introduction to Layout

The HTML id attribute

```
<p>Spatula City! Spatula City!</p> <p id="mission">Our mission is to provide the most spectacular spatulas and splurge on our specials until our customers <q>esplode</q> with splendor!</p>
```

HTML

Spatula City! Spatula City!
Our mission is to provide the most spectacular spatulas and splurge on our specials until our customers esplode with splendor!

output

- allows you to give a unique ID to any element on a page
- each ID must be unique; can only be used once in the page

Linking to sections of a Web page

```
<p>Visit <a href= “http://www.textpad.com/download/
index.html#downloads"> textpad.com</a> to get the TextPad
editor.</p> <p><a href="#mission">View our Mission Statement</
a></p>
```

HTML

Visit textpad.com to get the TextPad editor.
View our Mission Statement

output

- a link target can include an ID at the end, precede by a #
- browser will load that page and scroll to element with given ID

CSS ID selectors

```
#mission {
    font-style: italic;
    font-family: "Garamond", "Century Gothic", serif;
}
```
HTML

Spatula City! Spatula City!

Our mission is to provide the most spectacular spatulas and splurge on our specials until our customers esplode with splendor!

output

- applies style only to the paragraph that has the ID of mission
- element can be specified explicitly p#mission { … }

The HTML class attribute

```
<p class="shout">Spatula City! Spatula City!</p>
<p class="special">See our spectacular spatula specials!</p>
<p class="special">Today only: satisfaction guaranteed.</p>
```

HTML

Spatula City! Spatula City!
See our spectacular spatula specials!
Today only: satisfaction guaranteed.

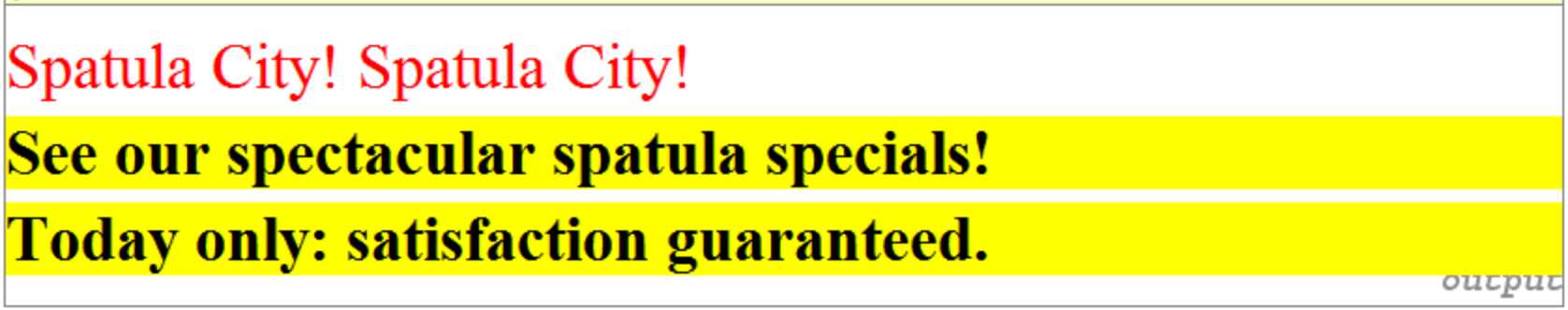
output

- classes are a way to group some elements and give a style to only that group
- unlike an id, a class can be reused as much as you like on the page

CSS class selectors

```css
.special {
  background-color: yellow;
  font-weight: bold;
}
p.shout {
  color: red;
  font-family: cursive;
}
```

CSS



output

- applies corresponding rule to any element with class special or a p with class shout

Multiple classes

```
<h2 class="shout">Spatula City!  Spatula City!</h2>
<p class="special">See our spectacular spatula specials!</p>
<p class="special shout">Satisfaction guaranteed.</p>
<p class="shout">We'll beat any advertised price!</p>
```

HTML

Spatula City! Spatula City!

See our spectacular spatula specials!

Satisfaction guaranteed.

We'll beat any advertised price!

output

- an element can be a member of multiple classes (separated by spaces)

Outline

- More CSS
- **Styling Page Sections**
- Introduction to Layout

Motivation for page sections

- want to be able to style **individual elements, groups of elements, sections of text** of the page
- (later) want to create complex page layouts

Sections of a page:

a section or division of your HTML page (block)

```
<div class="shout">
  <h2>Spatula City!  Spatula City!</h2>
  <p class="special">See our spectacular spatula specials!</p>
  <p>We'll beat any advertised price!</p>
</div>
```

HTML

Spatula City! Spatula City!

See our spectacular spatula specials!

We'll beat any advertised price!

output

- a tag used to indicate a logical section or area of a page
- has no appearance by default, but you can apply styles to it

Inline section:

an inline element used purely as a range for applying styles

```
<h2>Spatula City!  Spatula City!</h2>
<p>See our <span class="special">spectacular</span> spatula specials!</p>
<p>We'll beat <span class="shout">any advertised price</span>!</p>
```

HTML

Spatula City! Spatula City!

See our **spectacular** spatula specials!

We'll beat any advertised price!

output

- has no onscreen appearance, but you can apply a style or ID to it, which will be applied to the text inside the span
- So, when should we use , , and when <p>, <h1>, etc.?

CSS context selectors

```
selector1 selector2 {
    properties
}
```
CSS

- applies the given properties to *selector2* only if it is inside a *selector1* on the page

```
selector1 > selector2 {
    properties
}
```
CSS

- applies the given properties to *selector2* only if it is **directly** inside a *selector1* on the page (*selector2* tag is immediately inside *selector1* with no tags in between)

Context selector example

```html
<p>Shop at <strong>Hardwick's Hardware</strong>...</p>
<ul>
  <li>The <strong>best</strong> prices in town!</li>
  <li>Act while supplies last!</li>
</ul>
```

HTML

```css
li strong { text-decoration: underline; }
```

CSS

Shop at **Hardwick's Hardware**...

- The **best** prices in town!
- Act while supplies last!

output

More complex example

```html
<div id="ad">
  <p>Shop at <strong>Hardwick's Hardware</strong>...</p>
  <ul>
    <li class="important">The <strong>best</strong>
    prices in town!</li>
    <li>Act <strong>while supplies last!</strong></li>
  </ul>
</div>
```

HTML

```css
#ad li.important strong { text-decoration: underline; }
```

CSS

Shop at **Hardwick's Hardware**...

- The **best** prices in town!
- Act **while supplies last!**

output

CSS Inheritance

- Many properties in a CSS rule are inheritable by children elements of the rule specified, but some are not.
- types of properties are inheritable: text, color, and font
- types of properties are not: border, margin, padding
- all direct or inherited rules are same when considering their specificities
- if you can't remember whether a property is inheritable, better to figure it out by examining, other than via Googling or W3-Schooling

CSS Cascade

- Origin
  - the web page author
  - the browser user configuration
  - the browser default configuration
- Order
  - the order after all css rules inserted
- Specificity
  - a four digits number for each rule
- importancy
  - a rule is with or without `!important` modifier

Specificity of Selector

- the specificity of a CSS selector is a four digits number likes ***abcd***
- Count 1 if the styles are applied from the HTML style attribute, and 0 otherwise; this becomes variable ***a***.
- Count the number of ID attributes in the selector; the sum is variable ***b***.
- Count the number of attributes, pseudo-classes, and class names in a selector; the sum is variable ***c***.
- Count the number of element names in the selector; this is variable ***d***.
- Ignore pseudo-elements.
- when two conflict rules has same specificity, the one occurs later is style sheet file wins
- at last, a rule with **!important** overrides precedence!

```
body {
    font-size: 24px;
}
p {
    background: lightblue !important;
}
p {
    background: none;
}
```

The !important rule takes precedence.

Specificity of Selector

Selector	Selector Type	Specificity
*	Universal Selector	0000 (a = 0, b = 0, c = 0, d = 0)
li	Element Name	0001 (a = 0, b = 0, c = 0, d = 1)
ul li	Element Name	0002 (a = 0, b = 0, c = 0, d = 2)
div h1 + p	Element Name	0003 (a = 0, b = 0, c = 0, d = 3)
input[type='text']	Element Name + Attribute	0011 (a = 0, b = 0, c = 1, d = 1)
.someclass	Class Name	0010 (a = 0, b = 0, c = 1, d = 0)
div.someclass	Element Name + Class Name	0011 (a = 0, b = 0, c = 1, d = 1)
div.someclass.someother	Element Name + Class Name + Class Name	0021 (a = 0, b = 0, c = 2, d = 1)
#someid	ID Name	0100 (a = 0, b = 1, c = 0, d = 0)
div#someid	Element Name + ID Name	0101 (a = 0, b = 1, c = 0, d = 1)
style (attribute)	style (attribute)	1000 (a = 1, b = 0, c = 0, d = 0)

Rule conflicts algorithm

1. 所有适用于该元素的 CSS 规则将保持其在来源文件中的顺序，按照下面规则进行排序：
   a) 浏览器默认设置中的普通规则；
   b) 用户配置中的普通规则；
   c) 网页作者的普通规则；
   d) 网页作者的重要规则；
   e) 用户配置中的重要规则。
2. 计算每个规则的特异值，然后按照下述规则解决规则间的冲突：
   a) 特异值最高的重要规则胜出；
   b) 无重要规则，则特异值最高的普通规则胜出；
   c) 若两条规则重要性、特异值都相同，在第 1 步给出的来源排序中靠后的胜出；
   d) 若两条规则重要性、特异值、来源都相同，则在来源文件中按照 b)说明的规则顺序，后出现的胜出。

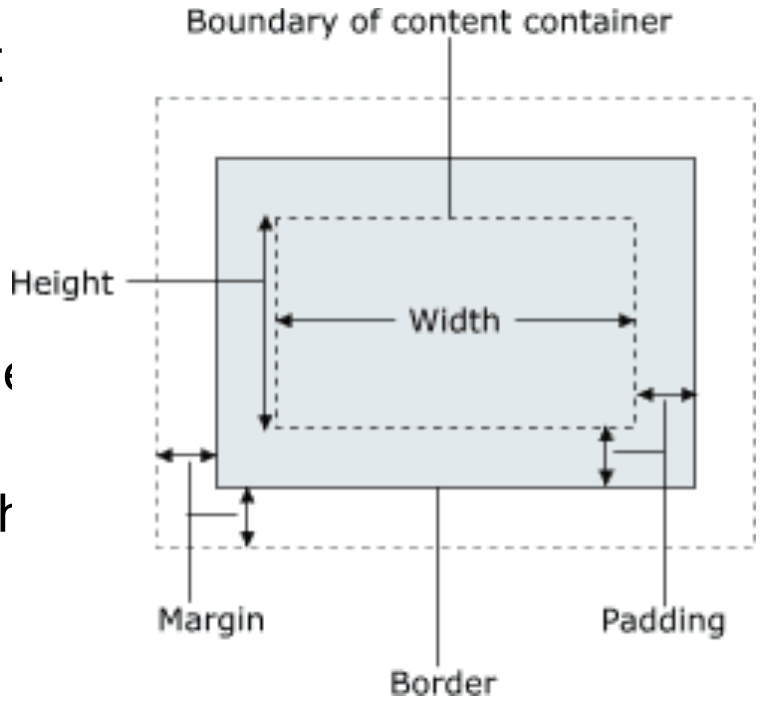
Outline

- More CSS
- Styling Page Sections
- **Introduction to Layout**

The CSS Box Model

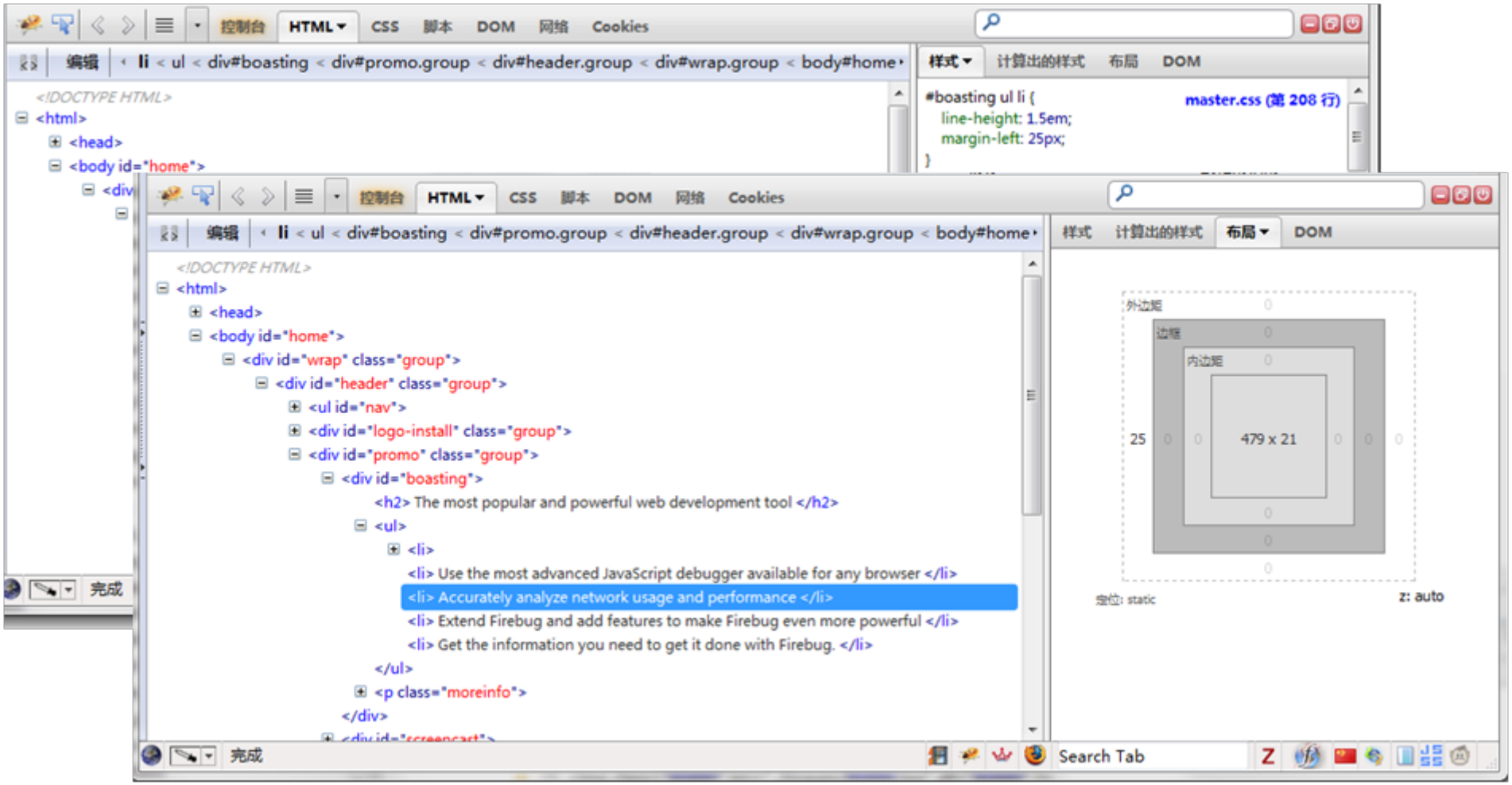
- for layout purpose, every element composed of:
  - the actual element's **content**
  - a **border** around the element
  - **padding** between the content and the border (*inside*)
  - a **margin** between the border and other content (outside)
- Width amount = width + L/R padding + L/R border + L/R margin

  height amount = content height + T/B padding + T/B border + T/B margin



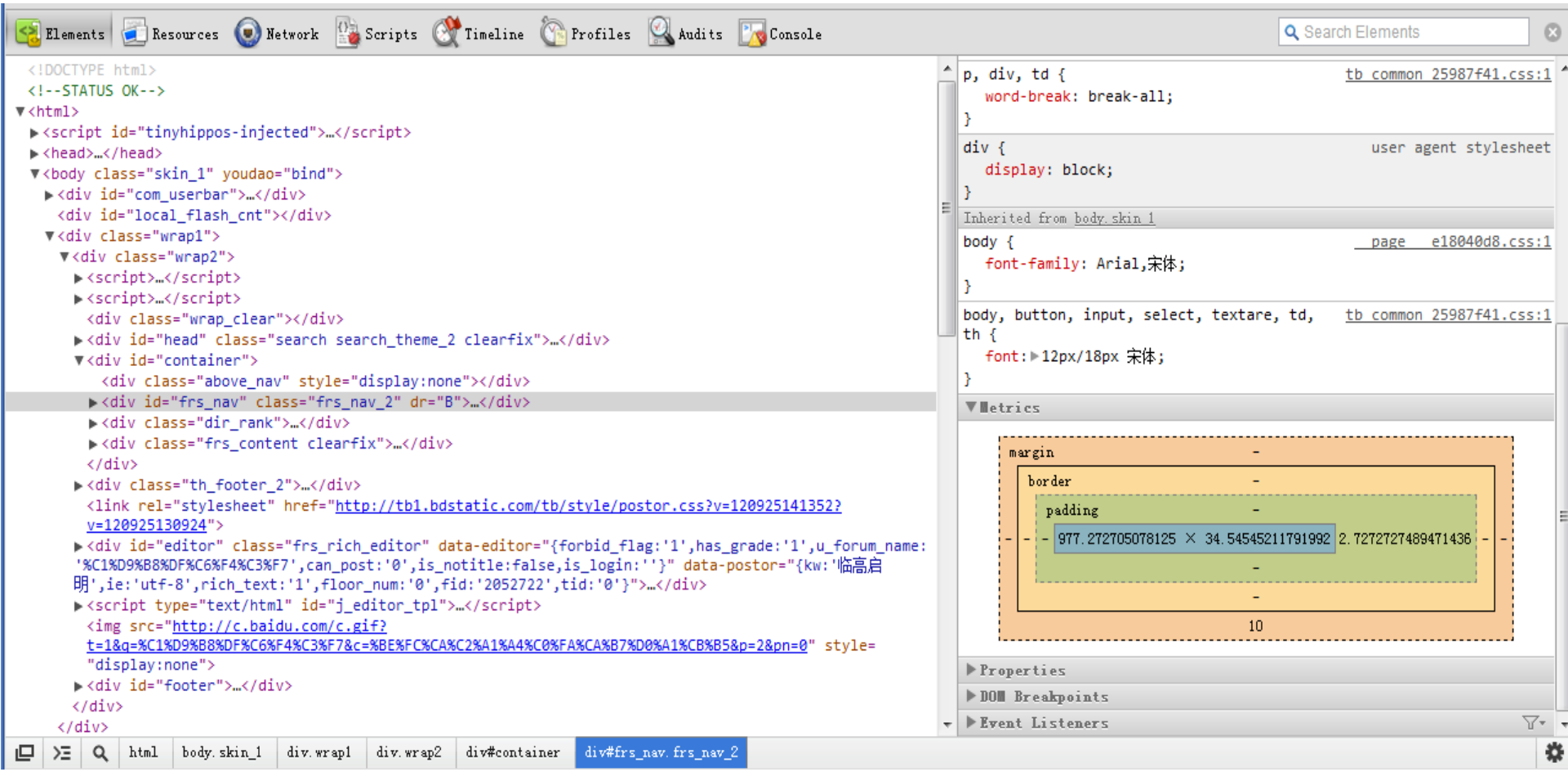
Debug Boxes of Web Page Elements

- Firefox



Debug Boxes of Web Page Elements

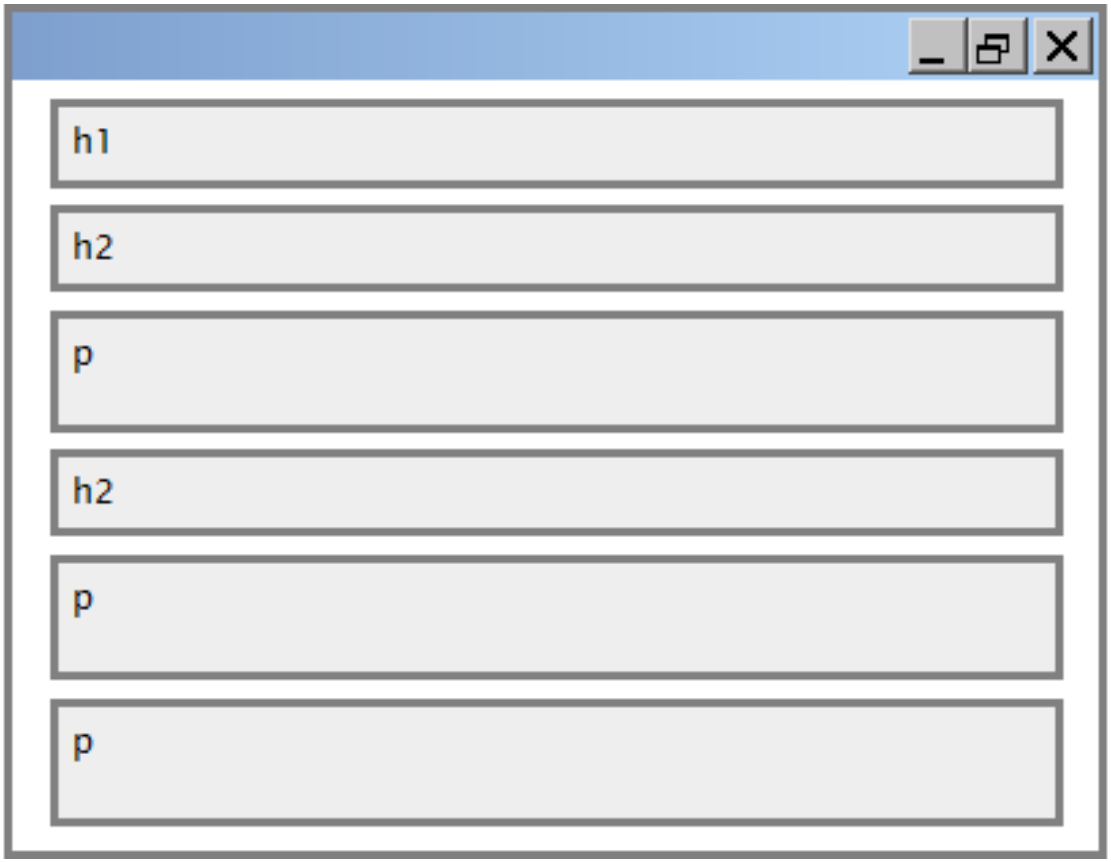
- Chrome



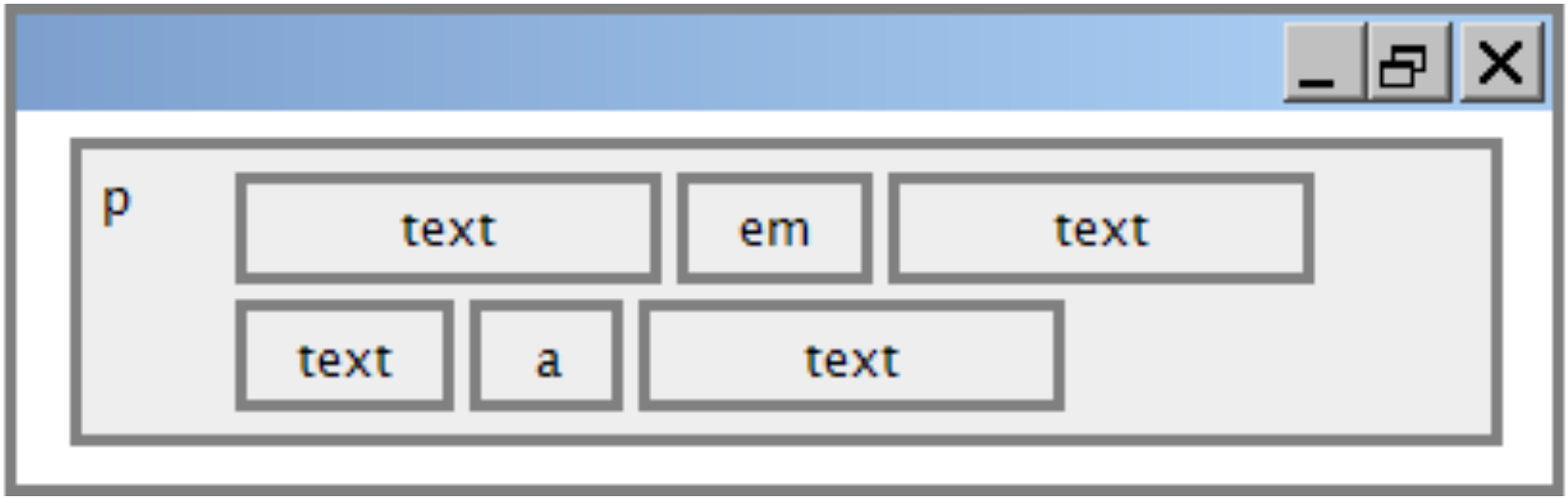
Debug Boxes of Web Page Elements

- IE

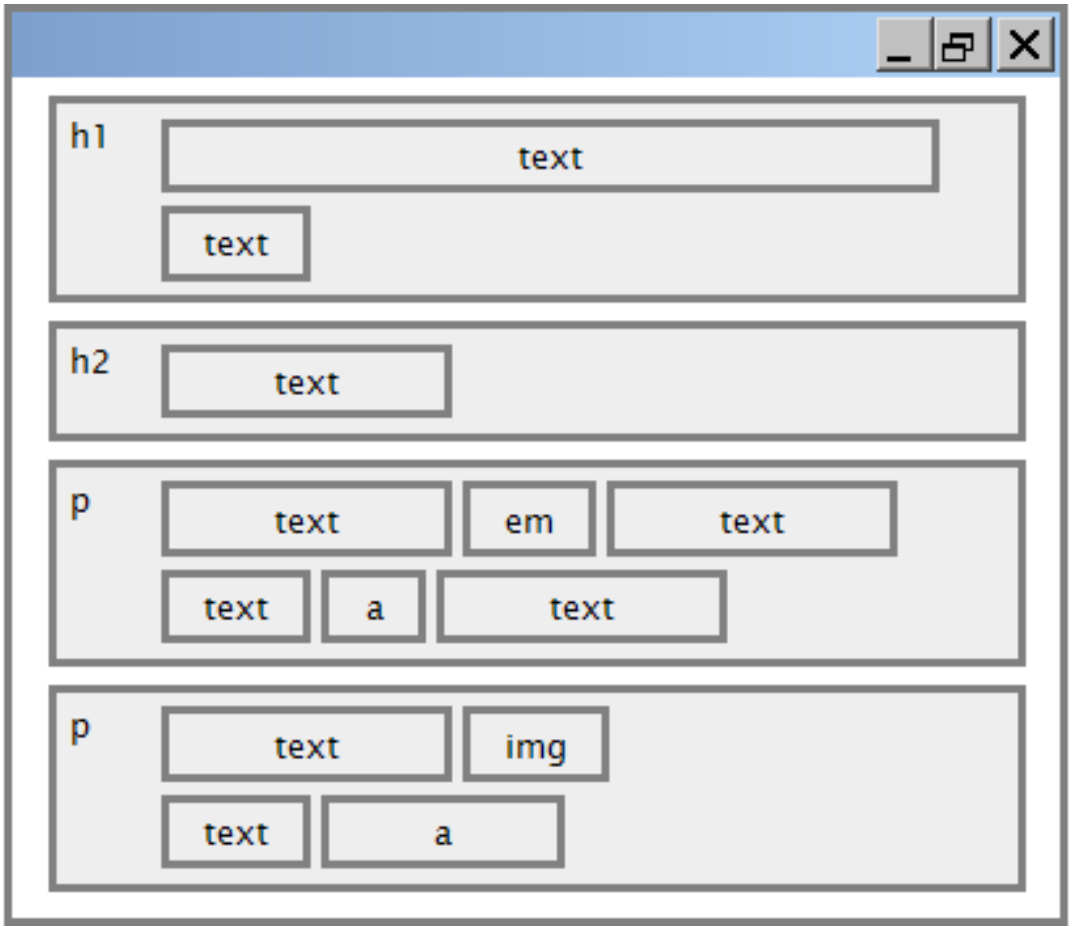
Document flow – block elements



Document flow – inline elements



Document flow – a larger example



CSS properties for borders

```css
h2 { border: 5px solid red; }
```

This is a heading.

property	description
border	thickness/style/size of border on all 4 sides

- **thickness** (specified in px, pt, em, or thin, medium, thick)
- **style** (none, hidden, dotted, dashed, double, groove, inset, outset, ridge, solid)
- **color** (specified as seen previously for text and background colors)

More border properties

property	description
border-color, border-width, border-style	specific properties of border on all 4 sides
border-bottom, border-left, border-right, border-top	all properties of border on a particular side
border-bottom-color, border-bottom-style, border-bottom-width, border-left-color, border-left-style, border-left-width, border-right-color, border-right-style, border-right-width, border-top-color, border-top-style, border-top-width	properties of border on a particular side
Complete list of border properties	

Border example 2

```css
h2 {
  border-left: thick dotted #CC0088;
  border-bottom-color: rgb(0, 128, 128);
  border-bottom-style: double;
}
```

CSS

This is a heading.

output

- each side's border properties can be set individually
- if you omit some properties, they receive default values (e.g. border-bottom-width above)

CSS properties for padding

property	description
padding	padding on all 4 sides
padding-bottom	padding on bottom side only
padding-left	padding on left side only
padding-right	padding on right side only
padding-top	padding on top side only
Complete list of padding properties	

Padding example 1

```css
p { padding: 20px; border: 3px solid black; }
h2 { padding: 0px; background-color: yellow; }
```

CSS

This is the first paragraph

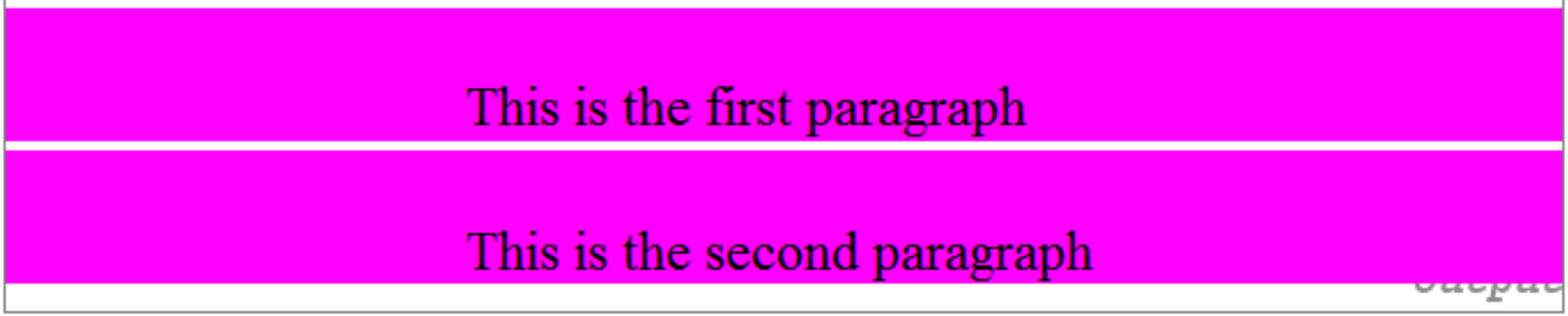
This is the second paragraph

This is a heading

Padding example 2

```css
p {
  padding-left: 200px; padding-top: 30px;
  background-color: fuchsia;
}
```

CSS



- each side's padding can be set individually
- notice that padding shares the background color of the element

CSS properties for margins

property	description
margin	margin on all 4 sides
margin-bottom	margin on bottom side only
margin-left	margin on left side only
margin-right	margin on right side only
margin-top	margin on top side only

Complete list of margin properties

Margin example 1

```css
p {
  margin: 50px;
  background-color: fuchsia;
}
```

CSS

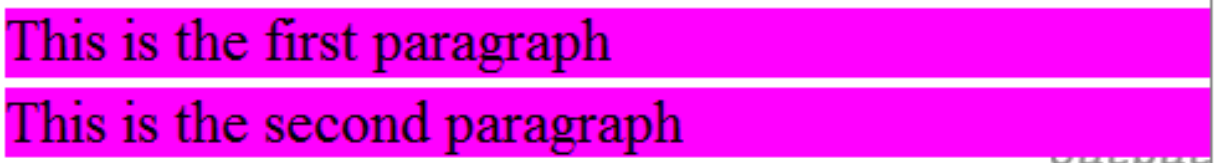
output

- notice that margins are always transparent
  (they don't contain the element's background color, etc.)

Margin example 2

```css
p {
  margin-left: 8em;
  background-color: fuchsia;
}
```

CSS



- each side's margin can be set individually

CSS properties for dimensions

```
p { width: 350px; background-color: yellow; }
h2 { width: 50%; background-color: aqua; }
```
CSS

This paragraph uses the first style above.

An h2 heading

output

property	description
width, height	how wide or tall to make this element (block elements only)
max-width, max-height, min-width, min-height	max/min size of this element in given dimension

Centering a block element: auto margins

```css
p {
  margin-left: auto;
  margin-right: auto;
  width: 750px;
}
```

CSS

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

output

- works best if width is set (otherwise, may occupy entire width of page)
- to center inline elements within a block element, use text-align: center

background-image

- Multiple background images

源代码 3-14 background-image 属性（多图片）示例

```
<td style='background-image:url(images/bird.png), url(images/泰山.jpg);'>多个图片背景 </td>
```

图 3-11 源代码 3-14 运行效果（左：运行效果；中：bird.png；右：泰山.jpg）

background-size

源代码 3-15 background-size 属性示例

```
<td>原始大小 </td>
<td style='background-size: 75px 120px'>75px 120px </td>
<td style='background-size: 120% 50%'>120% 50% </td>
<td style='background-size: cover'>cover </td>
<td style='background-size: contain'>contain </td>
```

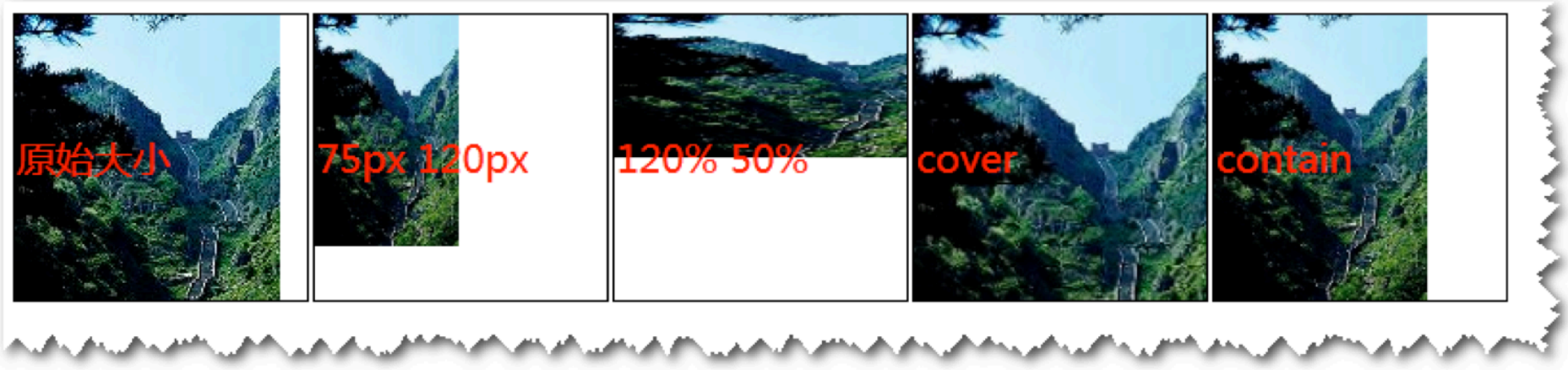


图 3-12 源代码 3-15 运行效果

background-position

源代码 3-16 background-position 属性示例

```
<td style='background-position: right top;'>右顶部<br/>（right top）</td>
<td style='background-position: bottom;'>底部（bottom）</td>
<td style='background-position: 25px 5px;'>距左上角<br/>（25，5）像素</td>
<td style='background-position: 50% 50%;'>50% 50%<br/>等价于center</td>
```

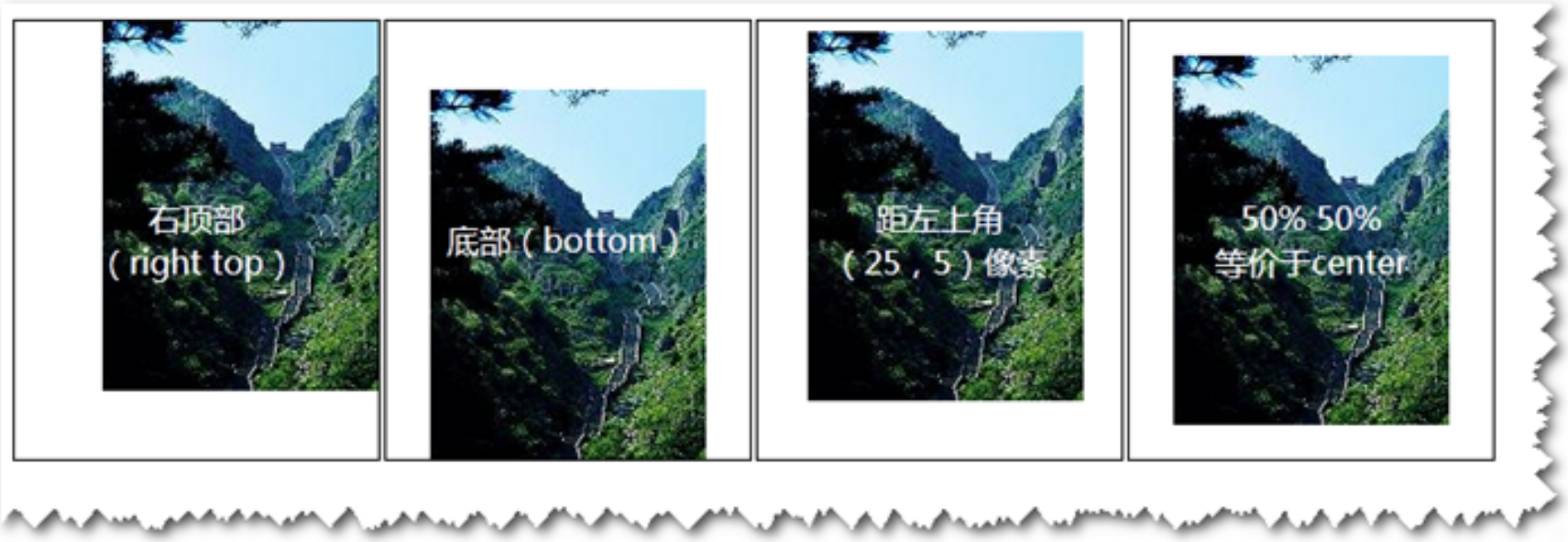


图 3-13 源代码 3-16 运行效果

background-origin

源代码 3-17 background-origin 属性示例

```
0   <style>
1       td{…… border: solid 10px rgba(255, 0, 0, 0.2); padding: 10px;}
2   </style>
3
4   <td>默认origin </td>
5   <td style='background-origin: padding-box'>padding-box </td>
6   <td style='background-origin: border-box'>border-box </td>
7   <td style='background-origin: content-box'>content-box </td>
```

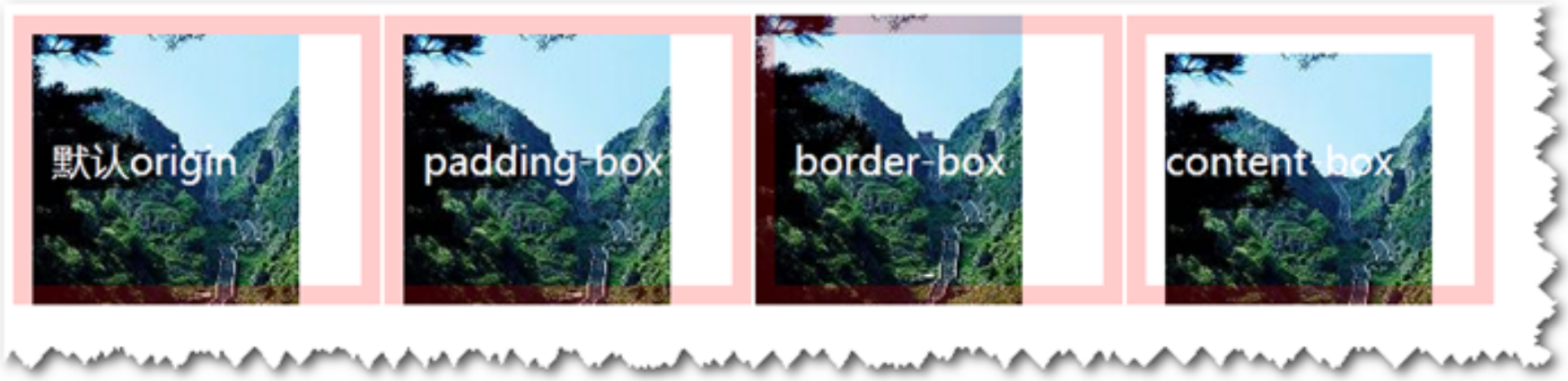


图 3-15 源代码 3-17 运行效果

background-clip

- 源代码 3-18 background-clip 属性示例

```
<style>
    td{
        padding: 10px; border: solid 10px rgba(255, 0, 0, 0.2);
        background-repeat:no-repeat;……
  }
</style>
<td>默认clip </td>
<td style='background-clip: padding-box'>padding-box </td>
<td style='background-clip: border-box'>border-box </td>
<td style='background-clip: content-box'>content-box </td>
```

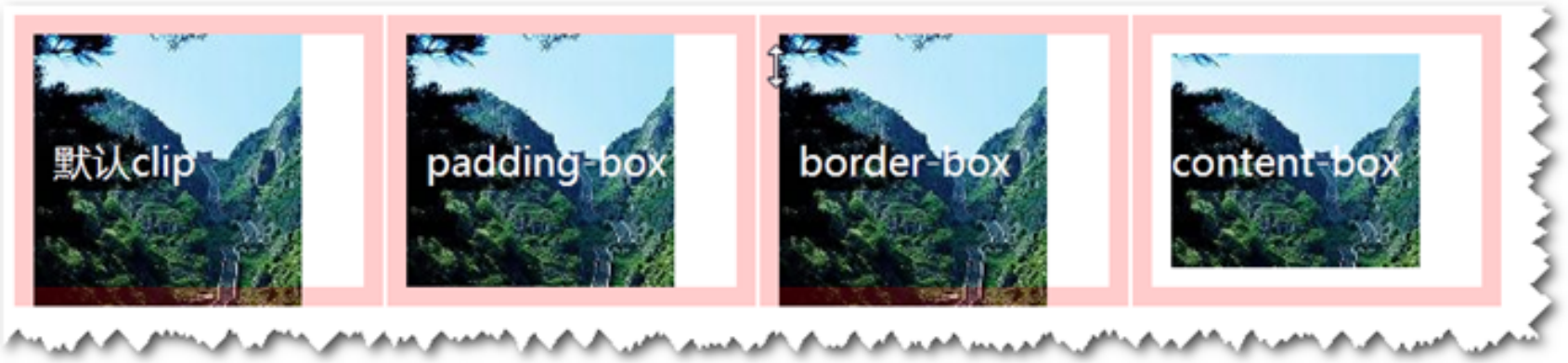


图 3-16 源代码 3-18 运行效果

background-repeat

- 源代码 3-19 background-repeat 属性示例

```
<td>从左上角开始<br/>同时横向、纵向重复</td>
<td style='background-repeat:repeat-y;'>从左上角开始<br/>纵向重复</td>
<td style='background-position: bottom; background-repeat:repeat-x;'>
从底部中间开始<br/>横向重复</td>
```

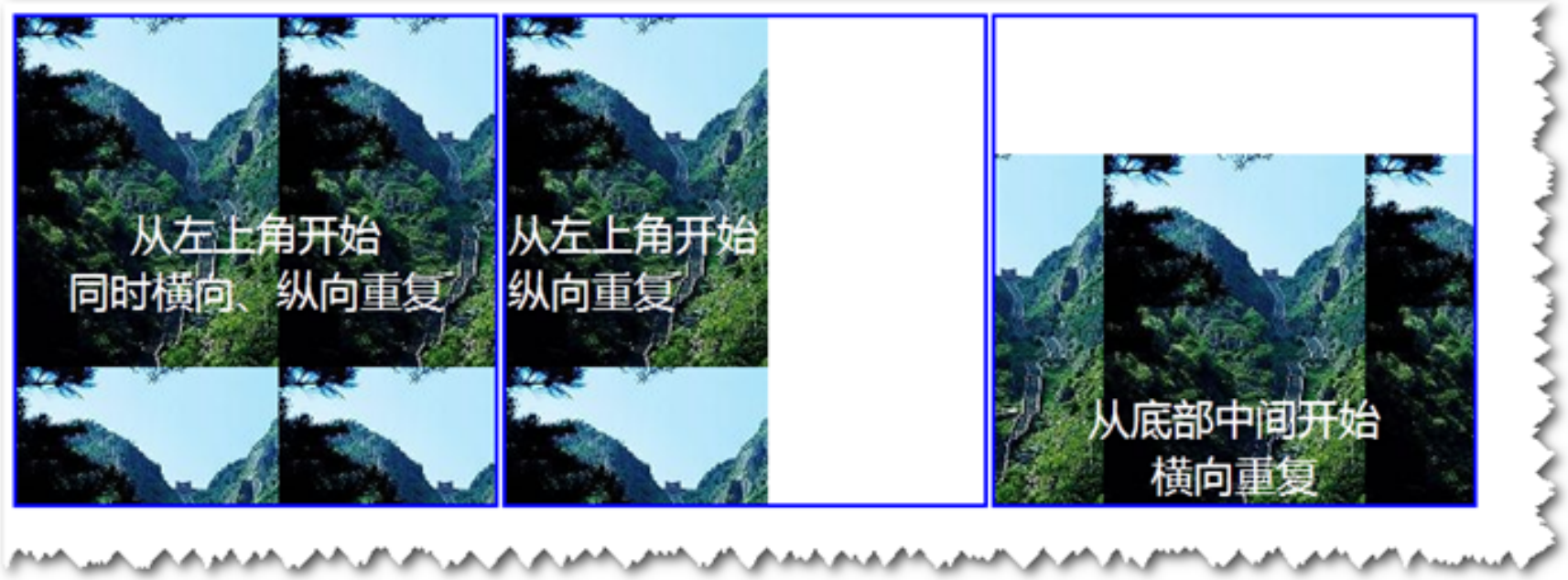


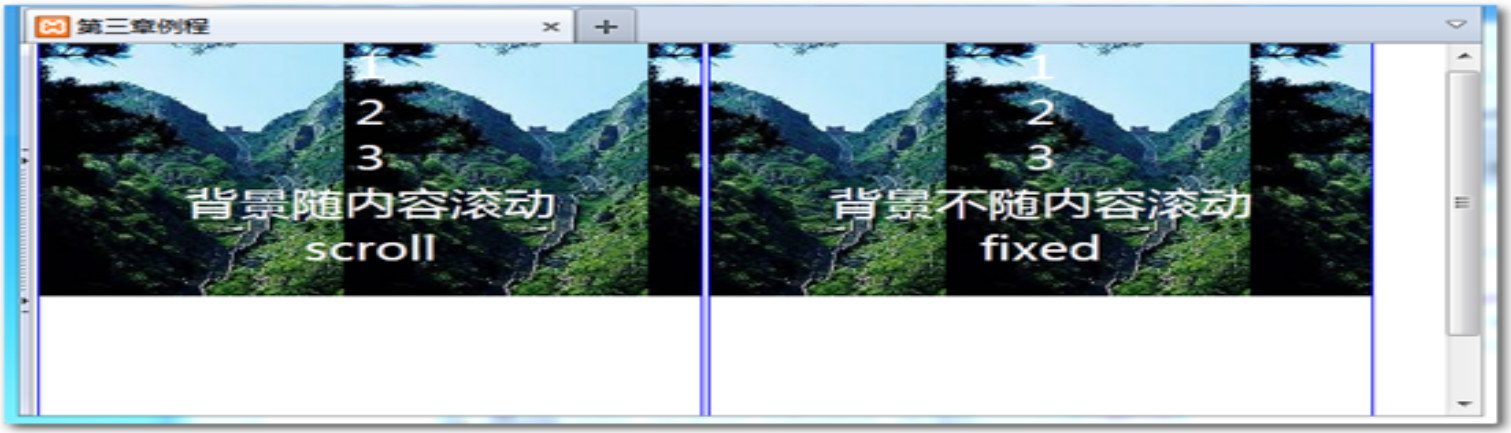
图 3-17 源代码 3-19 运行效果

background-attachment

源代码 3-20 background-attachment 属性示例

```
<style>
td { background-repeat:repeat-x;
……
}
</style>

<td>背景随内容滚动<br/>scroll</td>
<td style='background-attachment:fixed;'>背景不随内容滚动<br/>fixed</td>
```



Background all in one

源代码 3-21 background 属性示例

```
<td style='background-color: yellow; background-position: bottom
right; background-repeat:no-repeat; background-image:
url(images/泰山.jpg);'>分别表述</td>
<td style='background: no-repeat url(images/泰山.jpg) bottom right
yellow'>综合表述</td>
```

图 3-19 源代码 3-21 运行效果

Summary

- More CSS
  - HTML attributes: id, class
  - multiple classes
  - pseudo-classes
- Styling Page Sections
  - page section motivation
  - div, span
  - CSS context selector
  - CSS Cascading & Inheritance
- Introduction to Layout
  - Box Model, document flow
  - properies for borders, paddings, margins
  - properties for demensions

Exercises

- Complete our example of this class
  - the initial files can be downloaded from lecture_example
  - the result should like this:

Further Readings

- W3C CSS2 Specification: http://www.w3.org/TR/REC-CSS2/
- W3Schools CSS2 Reference: http://www.w3schools.com/css/css_reference.asp
- W3Schools CSS Tutorial: http://www.w3schools.com/css/default.asp
- Chapter 3, 4, 7, 8, and 11 of Beginning CSS Cascading Style Sheets for Web Design, second edition
- http://www.barelyfitz.com/screencast/html-training/css/positioning/
- http://www.quirksmode.org/css/display.html
- http://en.wikipedia.org/wiki/User-centered_design
- http://www.stcsig.org/usability/newsletter/9807-webguide.html

Part II

Outline

- **Floating Elements**
- Sizing and Positioning
- Thinking …
  - declarative programming
  - User Centric Design

The CSS float property

```
img.headericon {
  float: right;    width: 130px;
}
```

CSS

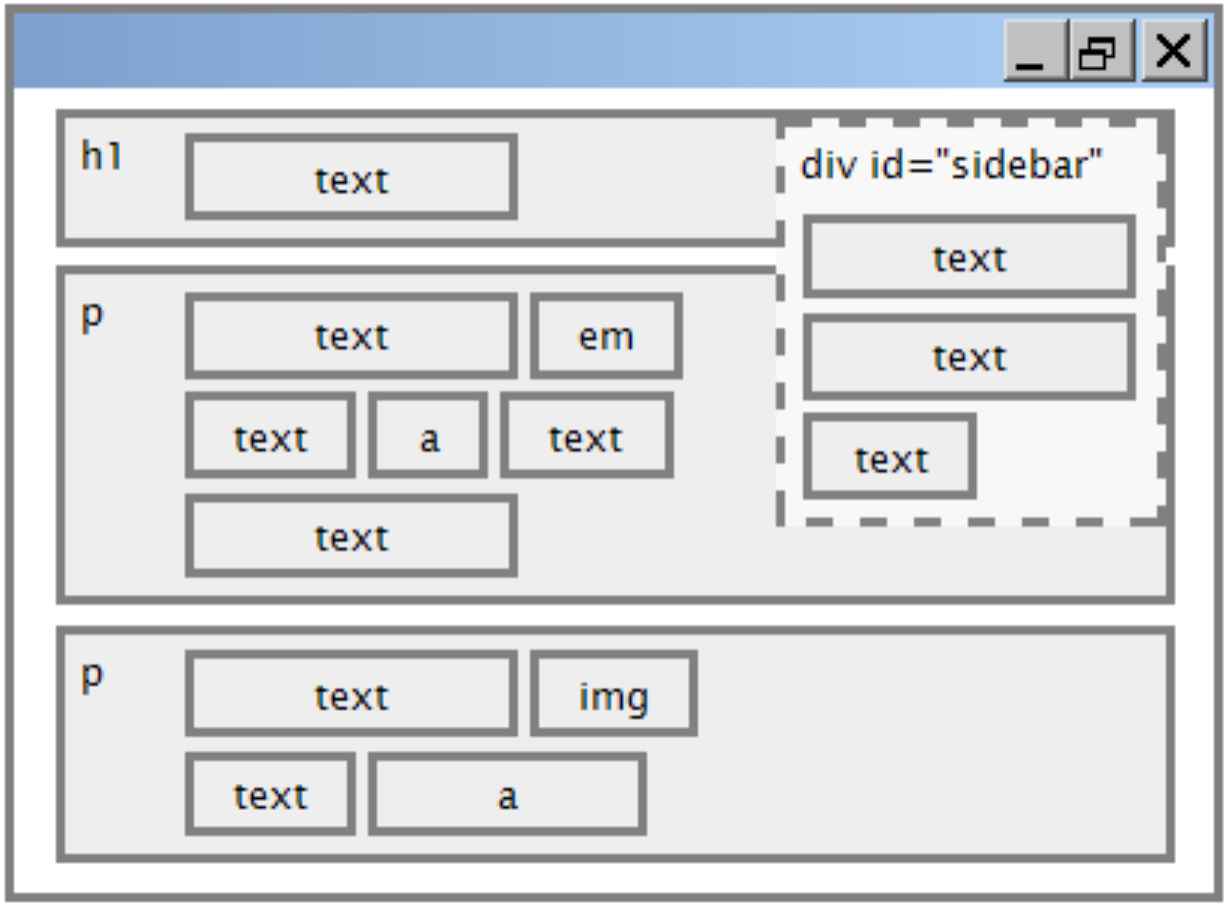
Borat Sagdiyev (born July 30, 1972) is a fictional Kazakhstani journalist played by British-Jewish comedian Sacha Baron Cohen. He is the main character portrayed in the controversial and successful film Borat: Cultural Learnings of America for Make Benefit Glorious ...

output

property	description
float	side to hover on; can be left, right, or none (default)

- removed from normal document flow; underlying text wraps around as necessary

Floating elements diagram



Common float bug: missing width

- often floating block elements must have a width property value
  - if no width is specified, the floating element may occupy 100% of the page width, so no content can wrap around it

The clear property

```
p { background-color: fuchsia; }
h2 { clear: right; background-color: yellow; }
```
CSS

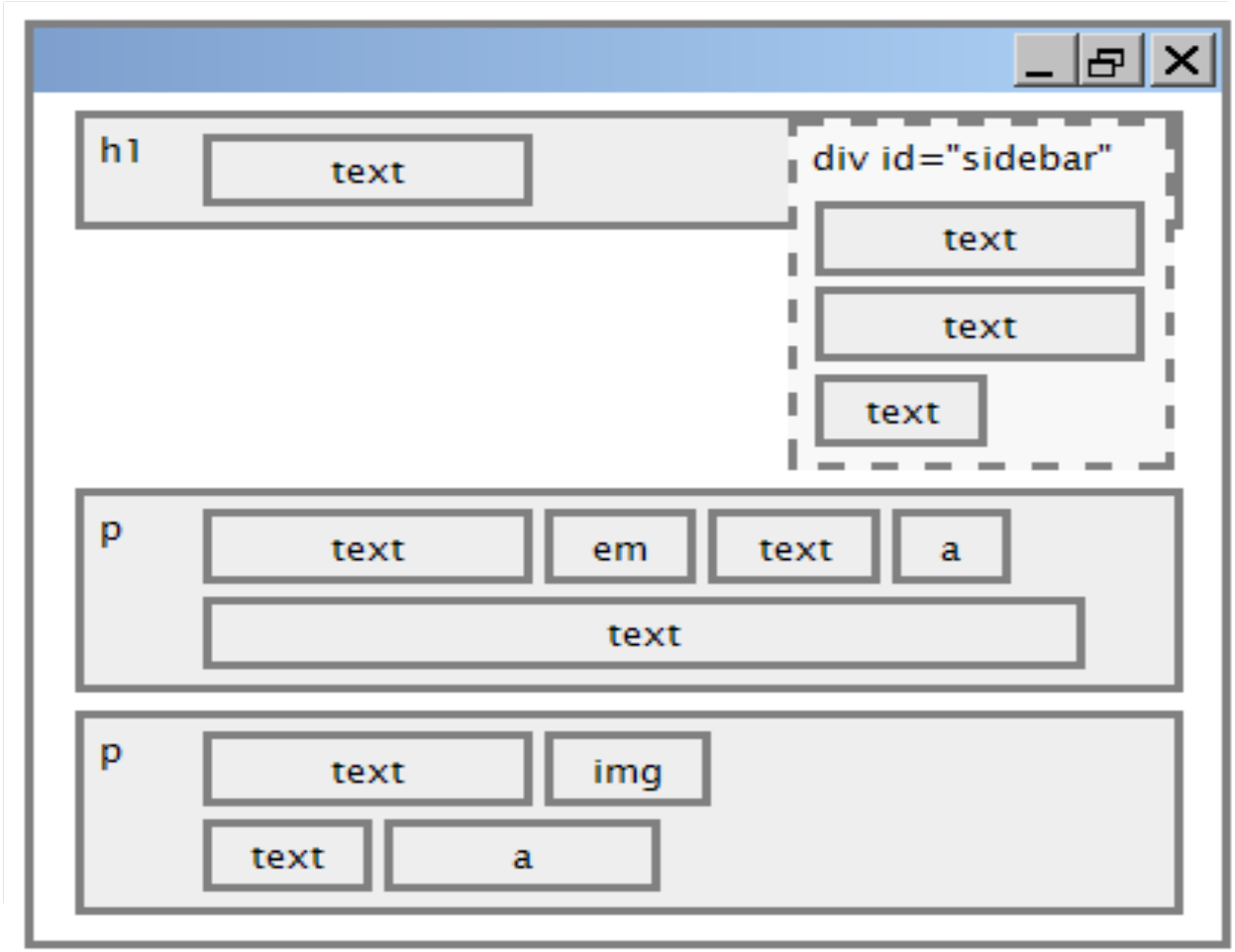
Homestar Runner is a Flash animated Internet cartoon. It mixes surreal humour with references to 1980s and 1990s pop culture, notably video games, classic television and popular music.

My Homestar Runner Fan Site

property	description
clear	disallows floating elements from overlapping this element; can be left, right, or none (default)

Clear diagram

```
div#sidebar { float: right; }
p { clear: right; }
```
CSS



Common error: container too short

```
<p><img src="images/homestar_runner.png" alt="homestar runner" />
  Homestar Runner is a Flash animated Internet cartoon.
  It mixes surreal humour with ....</p>
```
HTML

```
p { border: 2px dashed black; }
img { float: right; }
```
CSS

Homestar Runner is a Flash animated Internet cartoon. It mixes surreal humour with ....

- We want the `p` containing the image to extend downward so that its border encloses the entire image

The overflow property

```
p { border: 2px dashed black;
    overflow: hidden; }
```
CSS

Homestar Runner is a Flash animated Internet cartoon. It mixes surreal humour with ....

output

property	description
overflow	specifies what to do if an element's content is too large; can be auto, visible, hidden, scroll, or inherit

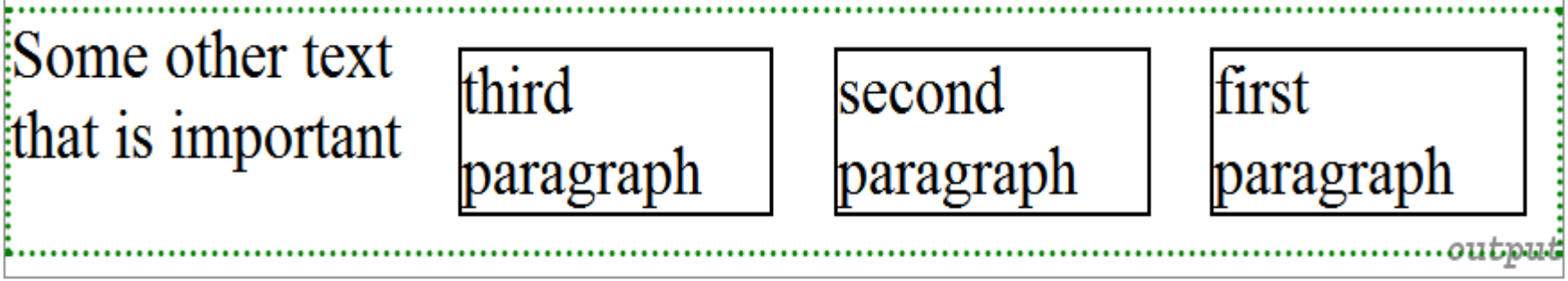
Multi-column layouts

```html
<div>
  <p>first paragraph</p>
  <p>second paragraph</p>
  <p>third paragraph</p>
  Some other text that is important
</div>
```

HTML

```css
p { float: right; width: 20%; margin: 0.5em;
    border: 2px solid black; }
div { border: 3px dotted green; overflow: hidden; }
```

CSS



output

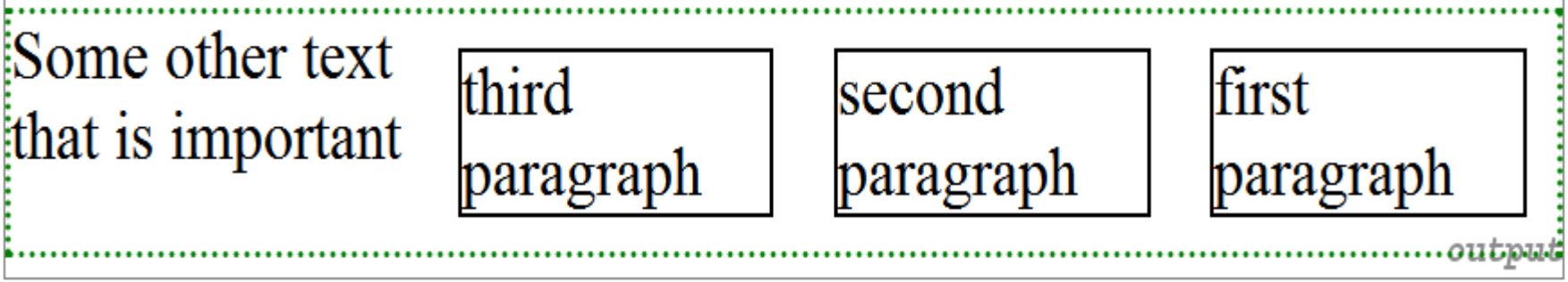
Multi-column layouts

```html
<div>
  <p>first paragraph</p>
  <p>second paragraph</p>
  <p>third paragraph</p>
  Some other text that is important
</div>
```

HTML

```css
p { float: right; width: 20%; margin: 0.5em;
    border: 2px solid black; }
div { border: 3px dotted green; overflow: hidden; }
```

CSS



output

Outline

- Floating Elements
- **Sizing and Positioning**
- Thinking …
  - declarative programming
  - User Centric Design

The position property

```css
div#ad {
    position: fixed;
    right: 10%;
    top: 45%;
}
```

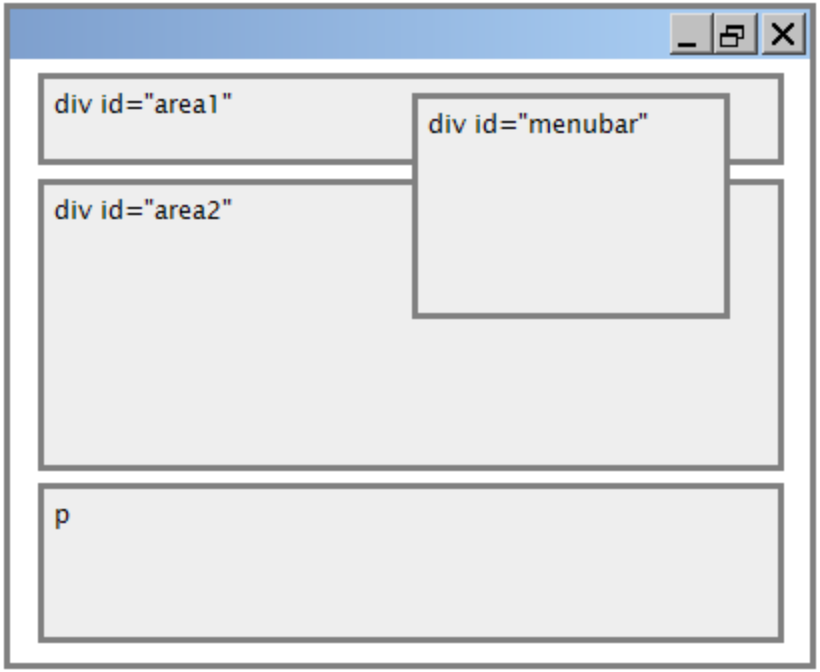
CSS

property	value	description
position	static	default position
	relative	offset from its normal static position
	absolute	a fixed position *within its containing element*
	fixed	a fixed position *within the browser window*
top, bottom, left, right	positions of box's corners	

Absolute positioning

```css
#menubar {
    position: absolute;
    left: 400px;
    top: 50px;
}
```
CSS

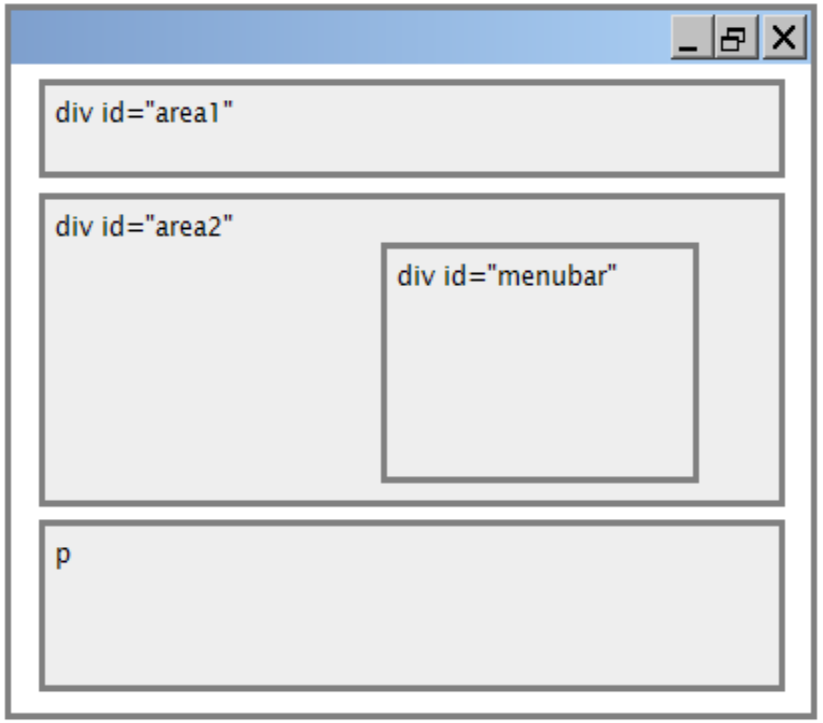
- removed from normal flow (like floating ones)
- positioned relative to the block element containing them (assuming that block also uses absolute or relative positioning)
- actual position determined by top, bottom, left, right values
- should often specify a width property as well



Relative positioning

```
#area2 { position: relative;}
```
CSS

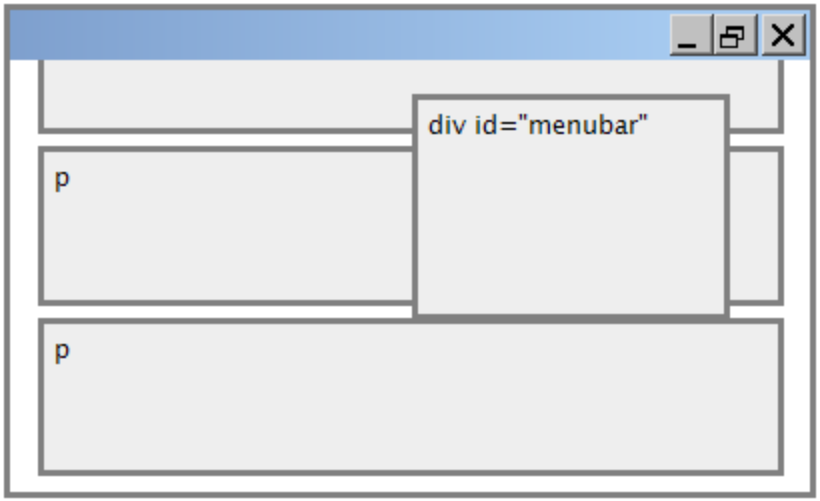
- absolute-positioned elements are normally positioned at an offset from the corner of the overall web page
- to instead cause the absolute element to position itself relative to some other element's corner, wrap the absolute element in an element whose position is relative



Fixed positioning

```css
#menubar {
    position: fixed;
    left: 400px;
    top: 50px;
}
```
CSS

- removed from normal flow (like floating ones)
- positioned relative to the browser window
  - even when the user scrolls the window, element will remain in the same place



Alignment vs. float vs. position

- if possible, lay out an element by ***aligning*** its content
  - horizontal alignment: text-align
    - set this on a block element; it aligns the content within it (not only text, and not the block element itself)
  - vertical alignment: vertical-align
    - set this on an inline element, and it aligns it vertically within its containing element
- if alignment won't work, try ***floating*** the element
- if floating won't work, try ***positioning*** the element
  - absolute / fixed positioning are a last resort and should not be overused
- more position examples

Details about inline boxes

- size properties (width, height, min-width, etc.) are ignored for inline boxes
- margin-top and margin-bottom are ignored, **but** margin-left and margin-right are **not**
- the containing block box's text-align property controls horizontal position of inline boxes within it
  - text-align does not align block boxes within the page
- each inline box's vertical-align property aligns it vertically within its block box

The vertical-align property

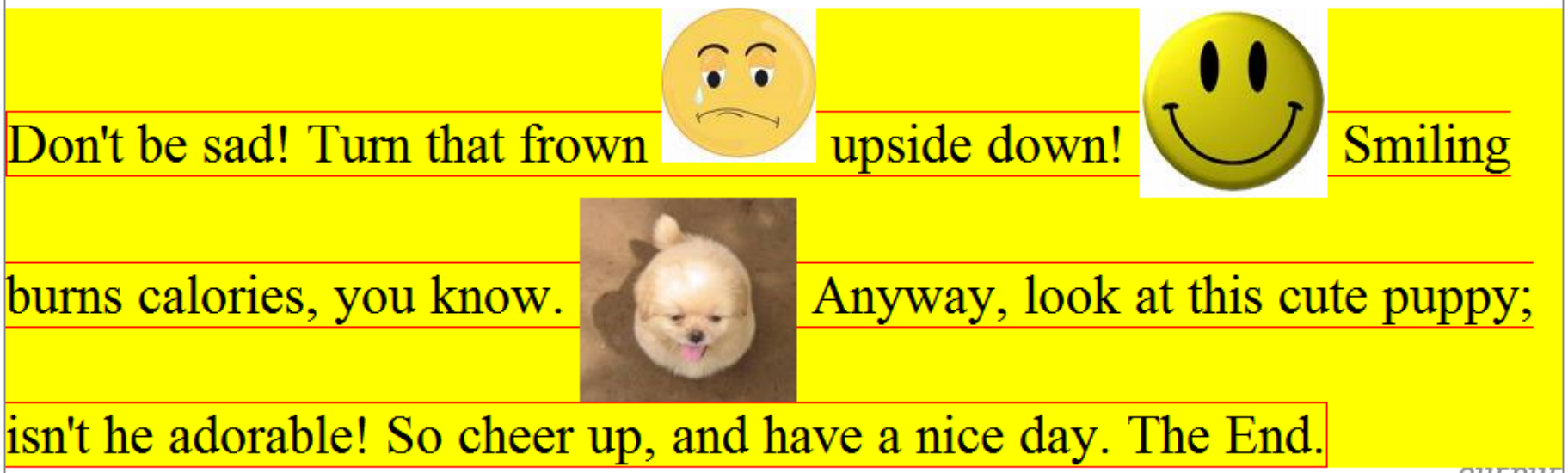
property	description
vertical-align	specifies where an inline element should be aligned vertically, with respect to other content on the same line within its block element's box

- can be top, middle, bottom, baseline (default), sub, super, text-top, text-bottom, or a length value or %
  - baseline means aligned with bottom of non -hanging letters

vertical-align example

```
<p style="background-color: yellow;">
<span style="vertical-align: top; border: 1px solid red;">
Don't be sad!  Turn that frown
<img src="images/sad.jpg" alt="sad" /> upside down!
<img style="vertical-align: bottom" src="images/smiley.jpg" alt="smile" />
Smiling burns calories, you know.
<img style="vertical-align: middle" src="images/puppy.jpg" alt="puppy" />
Anyway, look at this cute puppy; isn't he adorable!  So cheer up,
and have a nice day.  The End.
</span></p>
```

HTML



output

Common bug: space under image

```
<p style="background-color: red; padding: 0px; margin: 0px">
<img src="images/smiley.png" alt="smile" />
</p>
```

HTML

- red space under the image, despite padding and margin of 0
- this is because the image is vertically aligned to the baseline of the paragraph (not the same as the bottom)
- setting vertical-align to bottom fixes the problem (so does setting line-height to 0px)

The display property

```
h2 { display: inline; backgroud-color: yellow; }
```
CSS

This is a heading This is another heading

output

property	description
display	sets the type of CSS box model an element is displayed with

- values: none, inline, block, run-in, table, table-caption, …
  - not all values supported by all browsers (check out at http://www.quirksmode.org/css/display.html)
- use sparingly, because it can radically alter the page layout

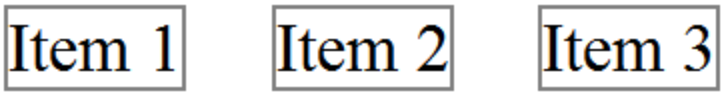
Displaying block element as inline

```html
<ul id="topmenu">
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>
```

HTML

```css
#topmenu li {
  display: inline;
  border: 2px solid gray;
  margin-right: 1em;
}
```

CSS



output

- lists and other block elements can be displayed inline
  - flow left-to-right on same line
  - width is determine by content (block elements are 100% of page width)

The visibility property

```
p.secret { visibility: hidden }
```
CSS

output

property	description
visibility	sets whether an element should be shown onscreen; can be visible (default) or hidden

- hidden elements will still take up space onscreen, but will not be shown
  - to make it not take up any space, set display to none instead
- can be used to show/hide HTML content on the page in response to events

Outline

- Floating Elements
- Sizing and Positioning
- **Thinking …**
  - **declarative programming**
  - User Centric Design

Declarative Programming

- **declarative programming** is a programming paradigm that expresses the logic of a computation without describing its control flow.
- DSL: SQL, CSS, HTML, WPDL, …
  - they are all common logics in software building
  - → extract common logics
  - → create a language describing them formally
  - → prove or verify the language
  - → use the language describe other logics
  - → alter the language to accommodate more scenarios
- Advantages of DSL -- externalized logics
  - easy coding & debugging
  - extendable & maintainable
  - reusable
  - …

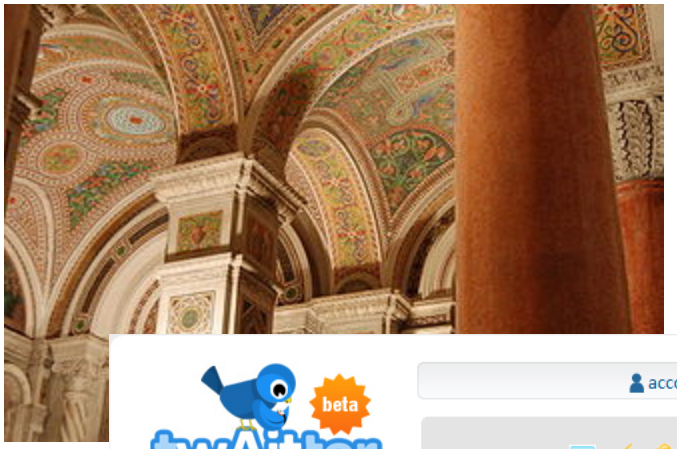
Outline

- Floating Elements
- Sizing and Positioning
- **Thinking …**
  - declarative programming
  - **User Centric Design**

What's the Design?

- **Design** is the planning that lays the basis for the making of every object or system.
  - As a verb, "to design" refers to the process of originating and developing a plan for a product, structure, system, or component with intention
  - As a noun, "a design" is used for either the final (solution) plan (e.g. proposal, drawing, model, description) or the result of implementing that plan in the form of the final product of a design process

What's a Design?

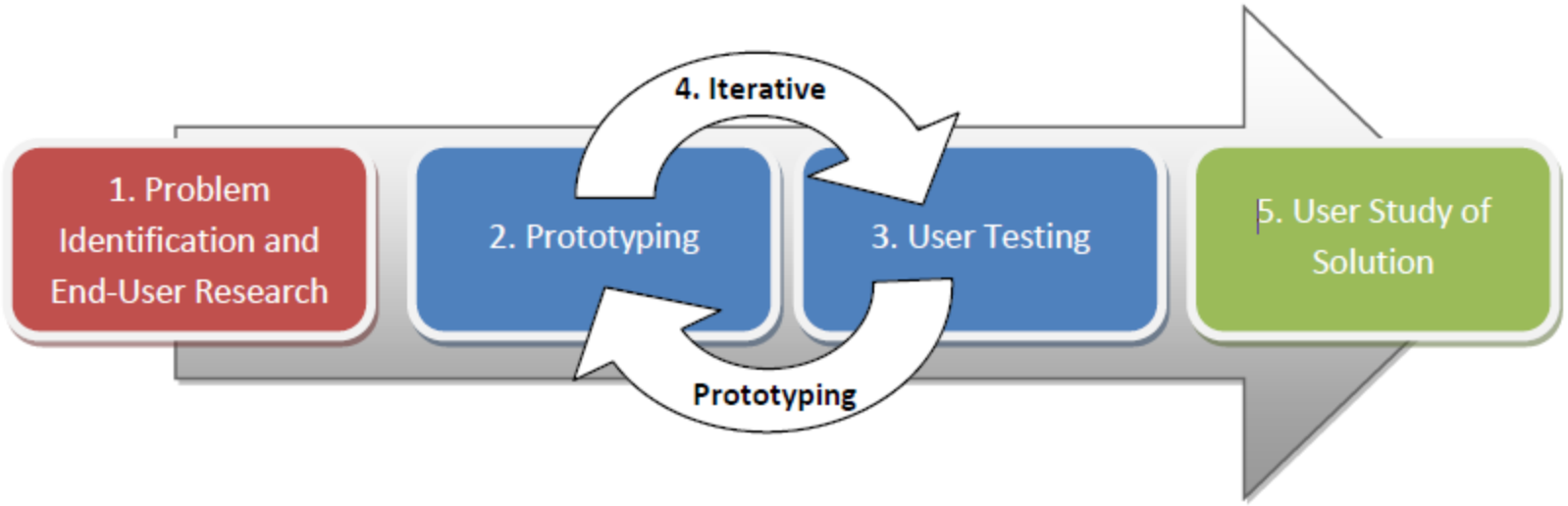


Design is about what we want, not how we get

User Centric Design

- In broad terms, **user-centered design (UCD)** is a design philosophy and a process in which the needs, wants, and limitations of end users of an interface or document are given extensive attention at each stage of the design process.
- User-centered design can be characterized as **a multi-stage problem solving process** that not only requires designers to **analyze and foresee** how users are likely to use an interface, but also to **test** the validity of their assumptions with regards to user behavior in real world tests with actual users.

Common UCD Process



- 1) Spend time with actual users or potential end-users to identify challenges they face, often with respect to a particular issue.
- 2) Prototype potential solutions.
- 3) User-test to see how the prototypes work or don't work.
- 4) Iteratively prototype and test, repeating steps 2 and 3.
- 5) Conduct a rigorous user study of your best solution. (Optional, but recommended)

UCD – Web Page: Purpose

- Who are the **users** of the Web page?
- What are the users' **tasks** and **goals**?
- What are the users' **experience** levels with the Web page, and Web page like it?
- What **functions** do the users need from the Web page?
- What **information** might the users need, and in what form do they need it?
- How do users think the Web page should work?

UCD – Web Page: Elements

- **Visibility**
  - mental model of the Web page
  - important elements should be emphatic
  - user should be able to tell from a glance what they can do and cannot do with the document
- **Accessibility**
  - users should be able to find information quickly and easily throughout the Web page (navigation, search, table of content, clear labeled sections, page numbers, color coding, etc.)
- **Legibility**
  - text should be easy to read (i.e. not too big or too small)
- **Language**
  - clear, active

UCD – Web Page: Rhetorical Situation

- **Audience**
  - people who will be using the document (age, geographical location, ethnicity, gender, education, etc.)
- **Purpose**
  - how the document will be used, and what the audience will be trying to accomplish while using the document (i.e. purchasing a product, selling ideas, performing a task, instruction, and all types of persuasion.)
- **Context**
  - the circumstances surrounding the situation.
    - What situation has prompted the need for this document?
    - Context also includes any social or cultural issues that may surround the situation.

Summary

- **Floating Elements**
  - float, clear, overflow
- **Sizing and Positioning**
  - position (absolute, relative, fixed)
  - alignment vs. float vs. position
  - inline boxes, vertical-align
  - display, visibility
- **Evil IE**
- **Declarative Programming – the life of DSL**
- **User Centric Design**
  - design, UCD
  - UCD process
  - UCD – Web page: purpose, elements, rhetorical situation

Exercises

- What are the most popular Web page fonts, and why?
- What are common layout elements of a contemporary Web page?
- Why “css + div” style layout is better than “table” style?
- Generally speaking, what’s the first step to build a Web site/app ?
- And how and by what means we are able to evaluate a design of a Web page, and which attributes of it are the most significant?

Further Readings

- W3C CSS2 Specification: http://www.w3.org/TR/REC-CSS2/
- W3 Schools CSS2 Reference: http://www.w3schools.com/css/css_reference.asp
- W3 Schools CSS Tutorial: http://www.w3schools.com/css/default.asp
- Chapter 3, 4, 7, 8, and 11 of Beginning CSS Cascading Style Sheets for Web Design, second edition (on Wiki)
- http://www.barelyfitz.com/screencast/html-training/css/positioning/
- http://www.quirksmode.org/css/display.html
- http://en.wikipedia.org/wiki/User-centered_design
- http://www.stcsig.org/usability/newsletter/9807-webguide.html

Thank you!