中山大學
SUN YAT-SEN UNIVERSITY

Lecture 5
Floating and Positioning

SE-805 Web 2.0 Programming

(http://my.ss.sysu.edu.cn/wiki/display/W2PSC/Home , **supported by Google;** *using some slides of & inspired by Marty Stepp's CSE 190 M courseware*)

School of Software, Sun Yat-sen University

Outline

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

The CSS float property

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

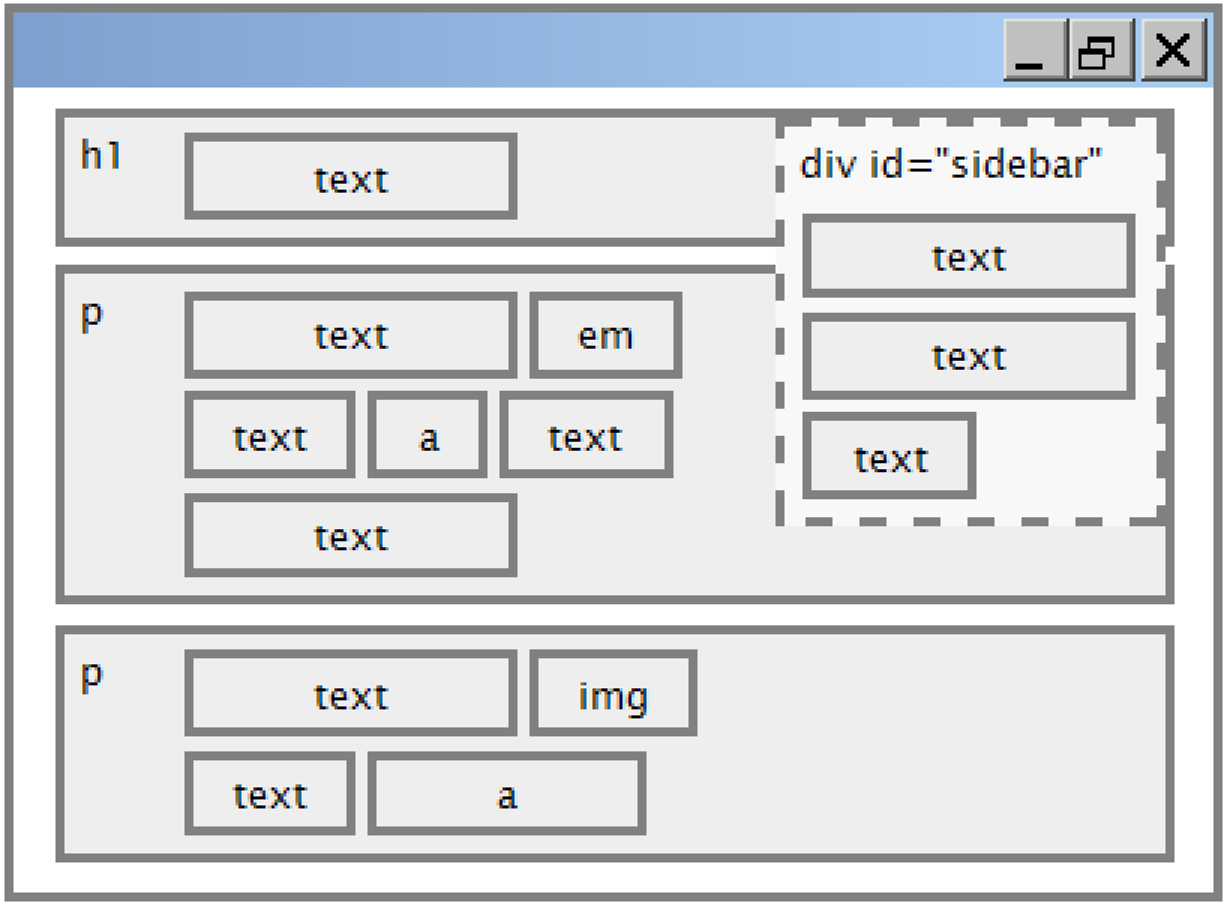
Borat Sagdiyev (born July 30, 1972) is a fictional Kazakhstani journalist played by British-Jewish comedian Sacha Baron Cohen. He is the main character portrayed in the controversial and successful film Borat: Cultural Learnings of America for Make Benefit Glorious ...

output

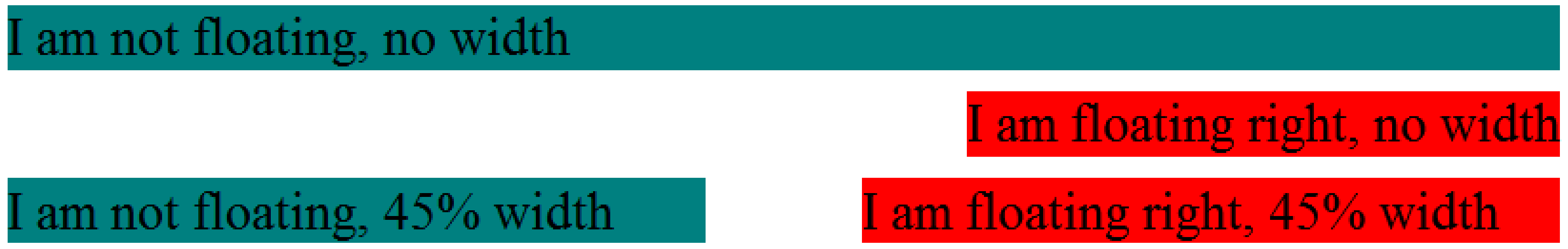
property	description
float	side to hover on; can be left, right, or none (default)

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

Floating elements diagram



Common float bug: missing width



I am not floating, no width

I am floating right, no width

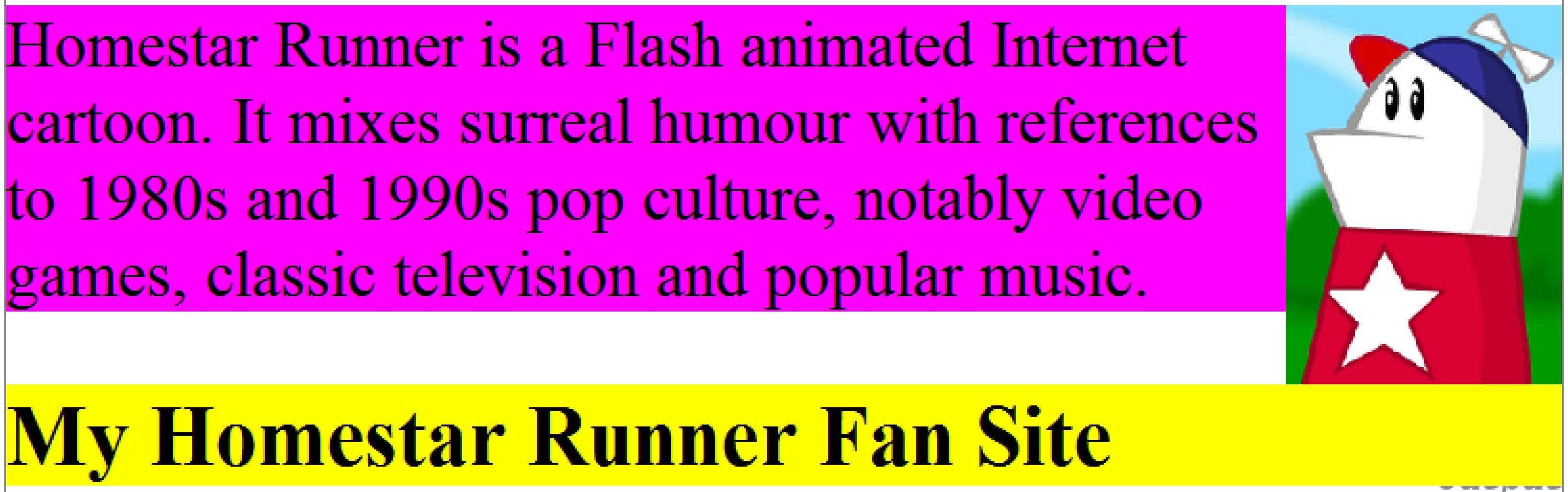
I am not floating, 45% width

I am floating right, 45% 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

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



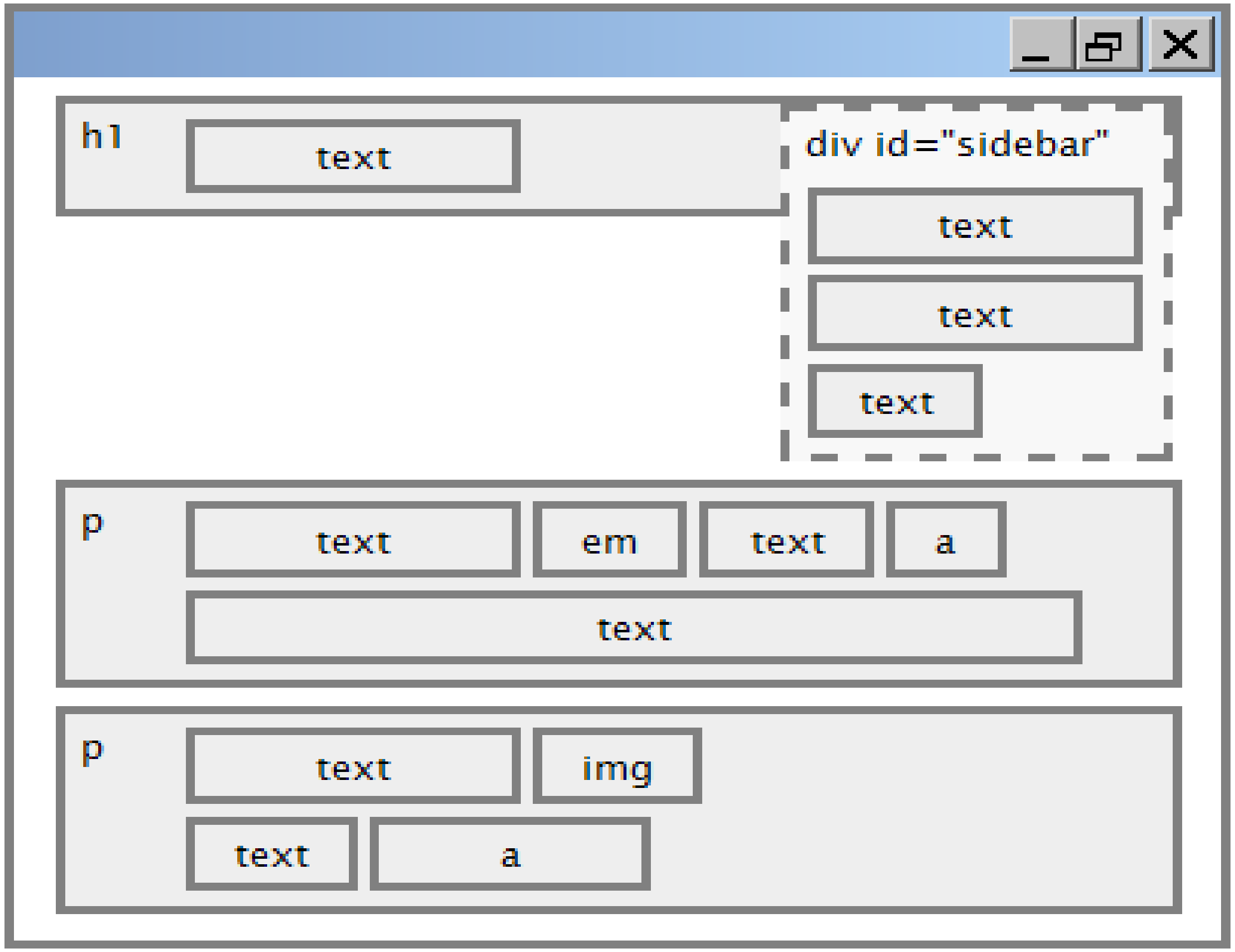
Homestar Runner is a Flash animated Internet cartoon. It mixes surreal humour with references to 1980s and 1990s pop culture, notably video games, classic television and popular music.

My Homestar Runner Fan Site

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

Clear diagram

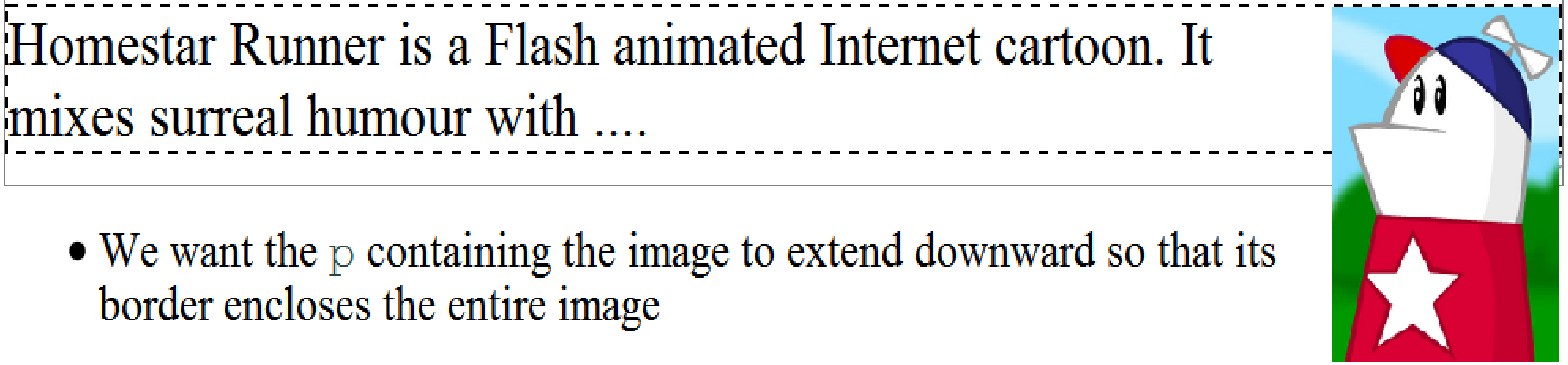
```css
div#sidebar { float: right; }
p { clear: right; }
```
CSS



Common error: container too short

```html
<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

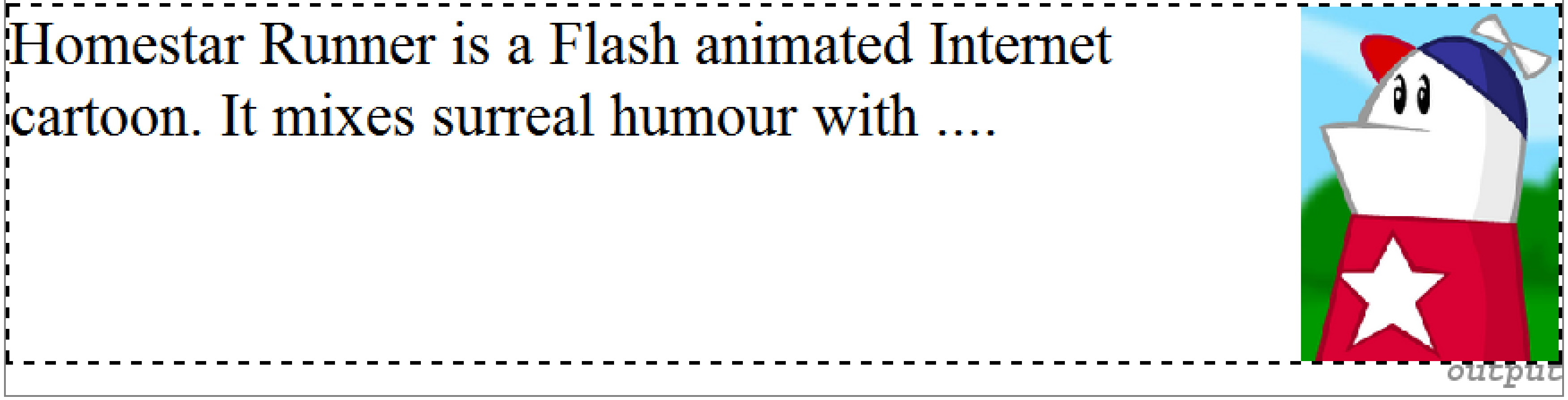
```css
p { border: 2px dashed black; }
img { float: right; }
```
CSS


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

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



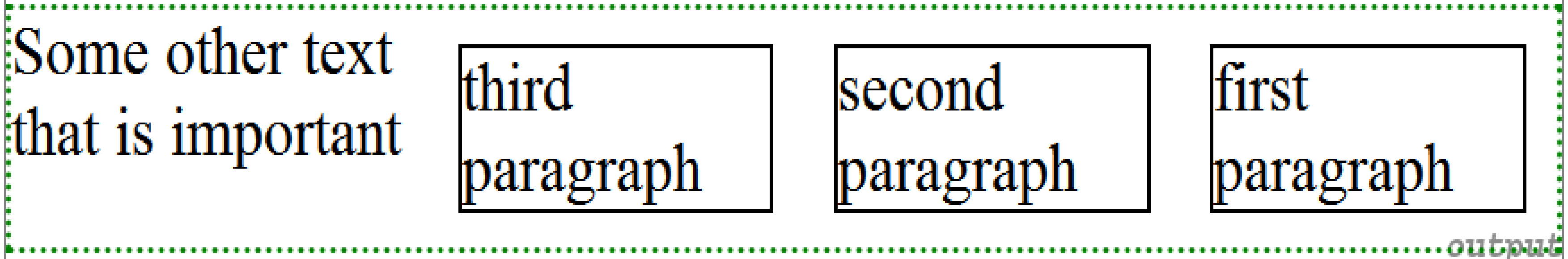
Homestar Runner is a Flash animated Internet cartoon. It mixes surreal humour with ....

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



Some other text that is important

third paragraph

second paragraph

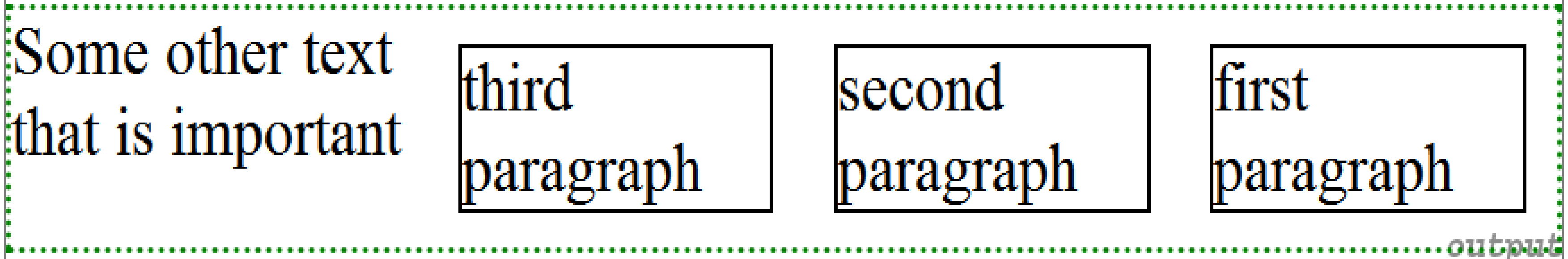
first paragraph

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


Some other text that is important

third paragraph

second paragraph

first paragraph

output

Outline

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

The position property

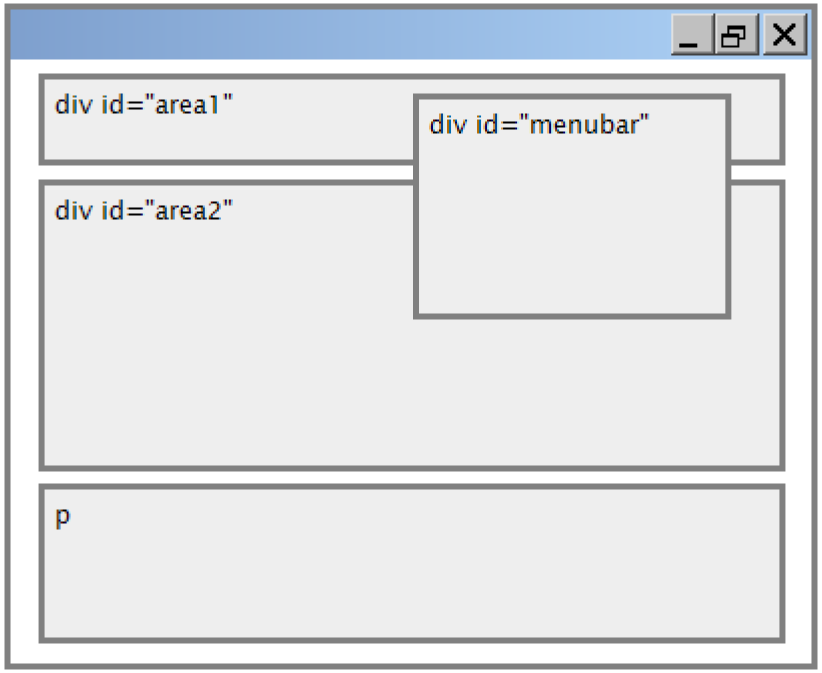
```
div#ad {
    position: fixed;
    right: 10%;
    top: 45%;
}
                                                    CSS
```

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

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

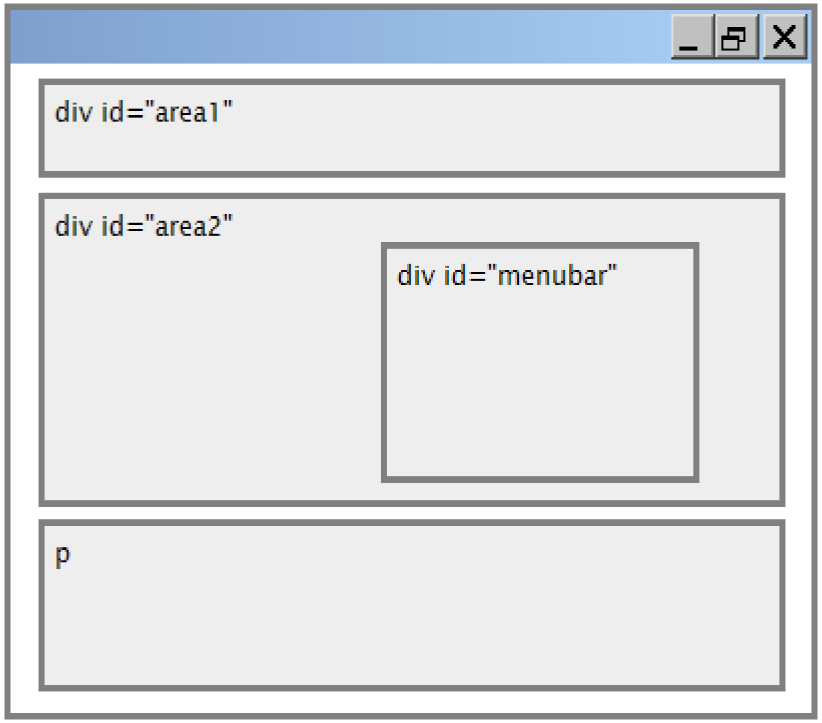
- removed from normal flow (like floating ones)
- positioned relative to the block element containing them (assuming that block also uses absolute or relative positioning)
- actual position determined by top, bottom, left, right values
- should often specify a width property as well



Relative positioning

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

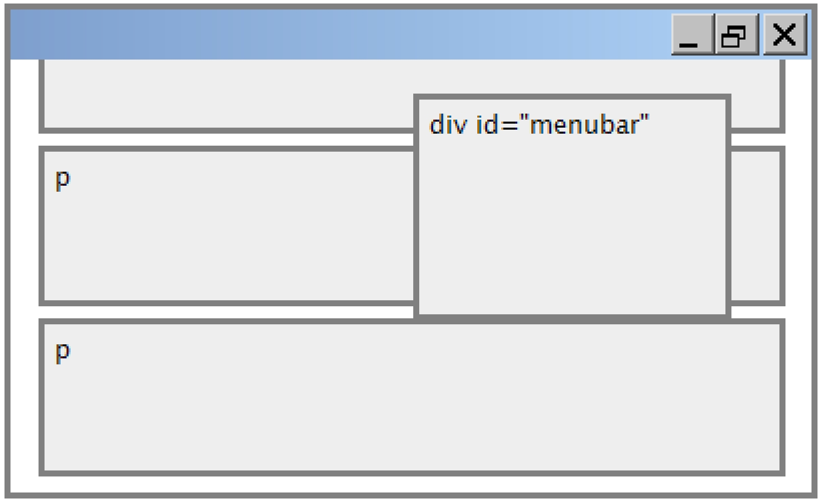
- absolute-positioned elements are normally positioned at an offset from the corner of the overall web page
- to instead cause the absolute element to position itself relative to some other element's corner, wrap the absolute element in an element whose position is relative



Fixed positioning

```
#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

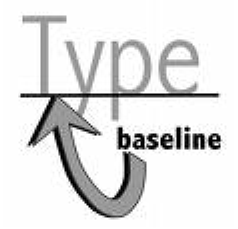
Details about inline boxes

- 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

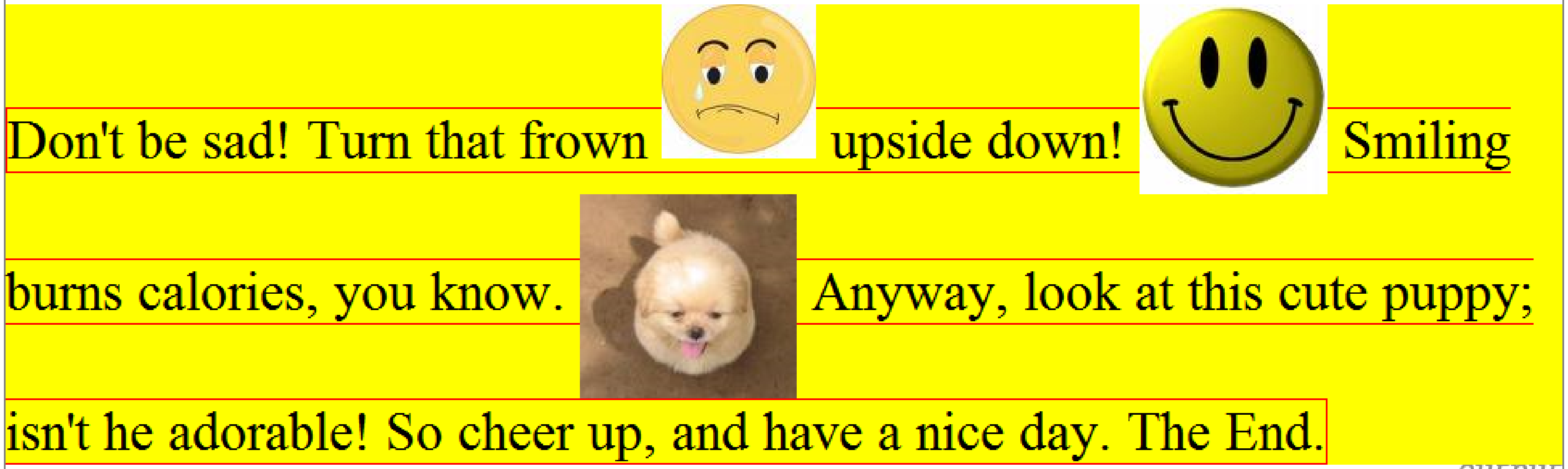
property	description
vertical-align	specifies where an inline element should be aligned vertically, with respect to other content on the same line within its block element's box

- 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



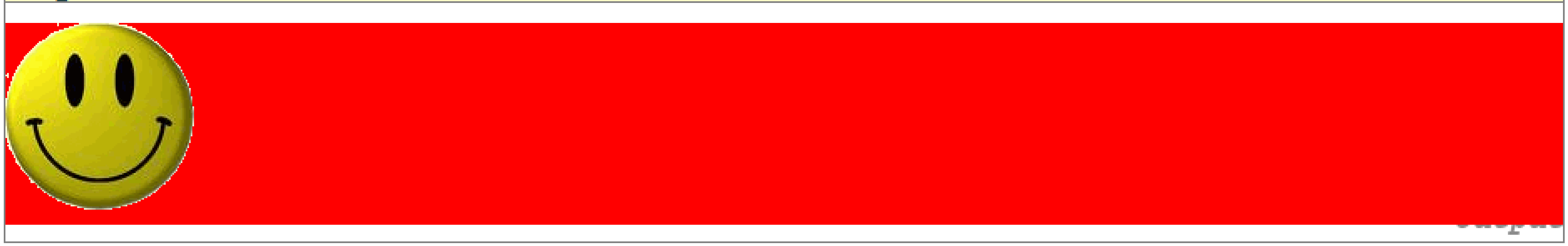
Don't be sad! Turn that frown upside down! Smiling burns calories, you know. Anyway, look at this cute puppy; isn't he adorable! So cheer up, and have a nice day. The End.

output

Common bug: space under image

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

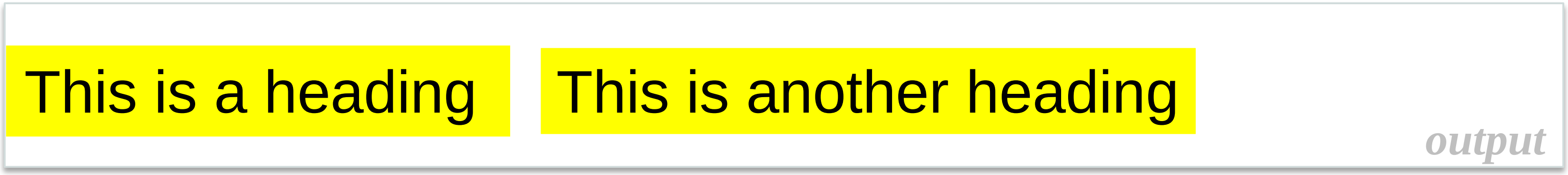
HTML



- 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

```css
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

```
Item 1   Item 2   Item 3
```
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

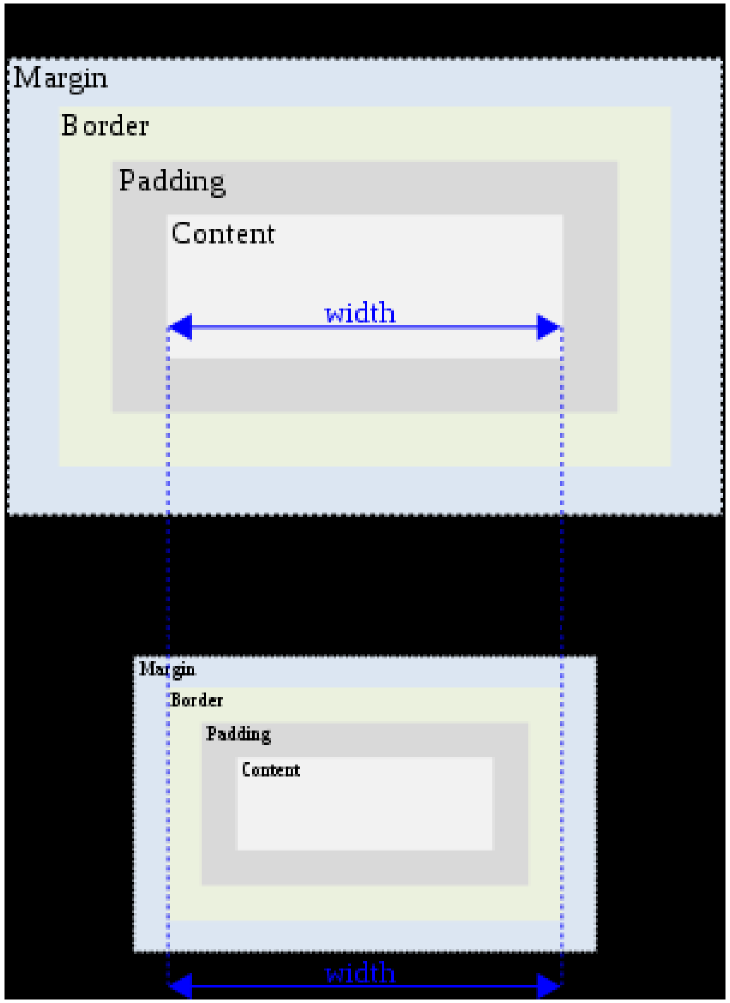
- hidden elements will still take up space onscreen, but will not be shown
  - to make it not take up any space, set display to none instead
- can be used to show/hide HTML content on the page in response to events

Outline

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

Evil IE

- IE is painful for Web designer and developer, since it doesn't compatible with W3C standards, and mostly deliberately …
- weird IE Box model
- double margin with float
- block has width floats when beneath a float element
- transparent png (IE 6.0)



Evil IE

- a lot of workarounds available, but the best is loading a specific style sheet for IE with conditional comment

```
<!--[if IE 7]>
   According to the conditional comment this is Internet Explorer
<![endif]-->
<!--[if gte IE 5]>
  According to the conditional comment this is Internet Explorer 5 and up
<![endif]-->
```

XHTML

- gt, lt, gte, lte

Outline

- Floating Elements
- Sizing and Positioning
- Evil IE
- **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
  - …

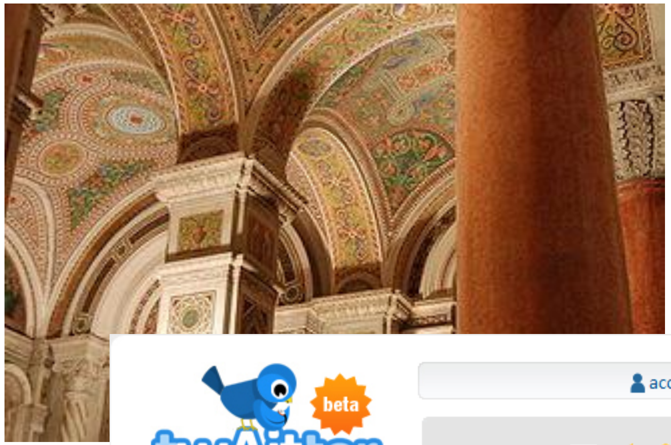
Outline

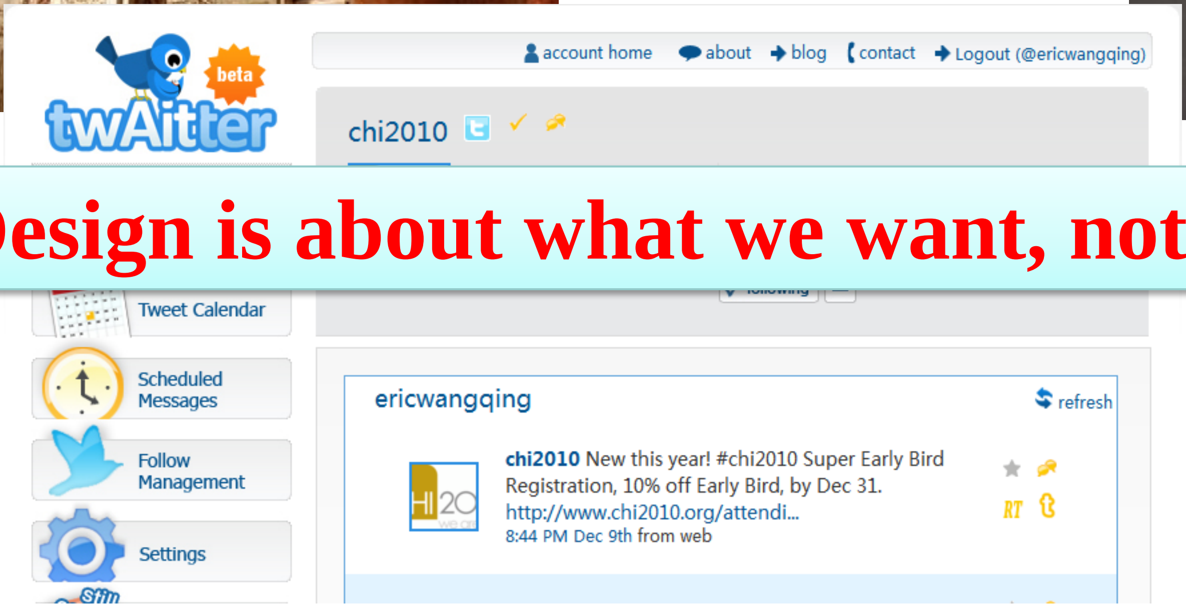
- Floating Elements
- Sizing and Positioning
- Evil IE
- **Thinking …**
  - declarative programming
  - **User Centric Design**

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?





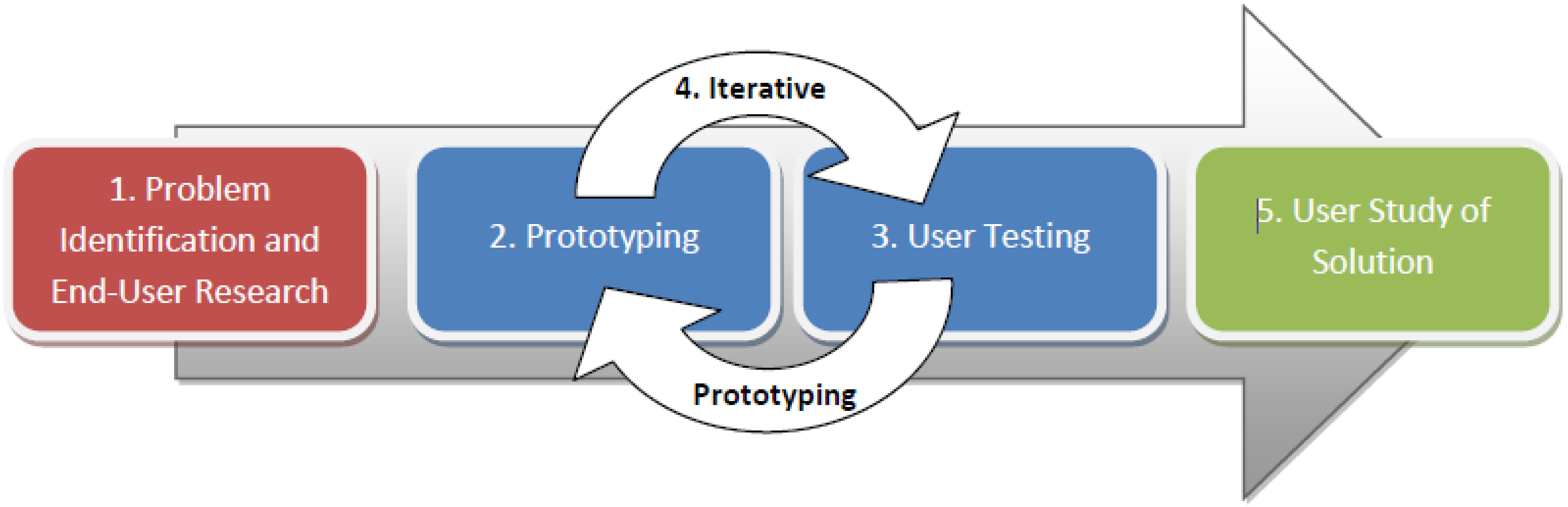
chi2010 New this year! #chi2010 Super Early Bird Registration, 10% off Early Bird, by Dec 31. http://www.chi2010.org/attendi...
8:44 PM Dec 9th from web

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. Problem Identification and End-User Research
2. Prototyping
4. Iterative
Prototyping
3. User Testing
5. User Study of Solution

- 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

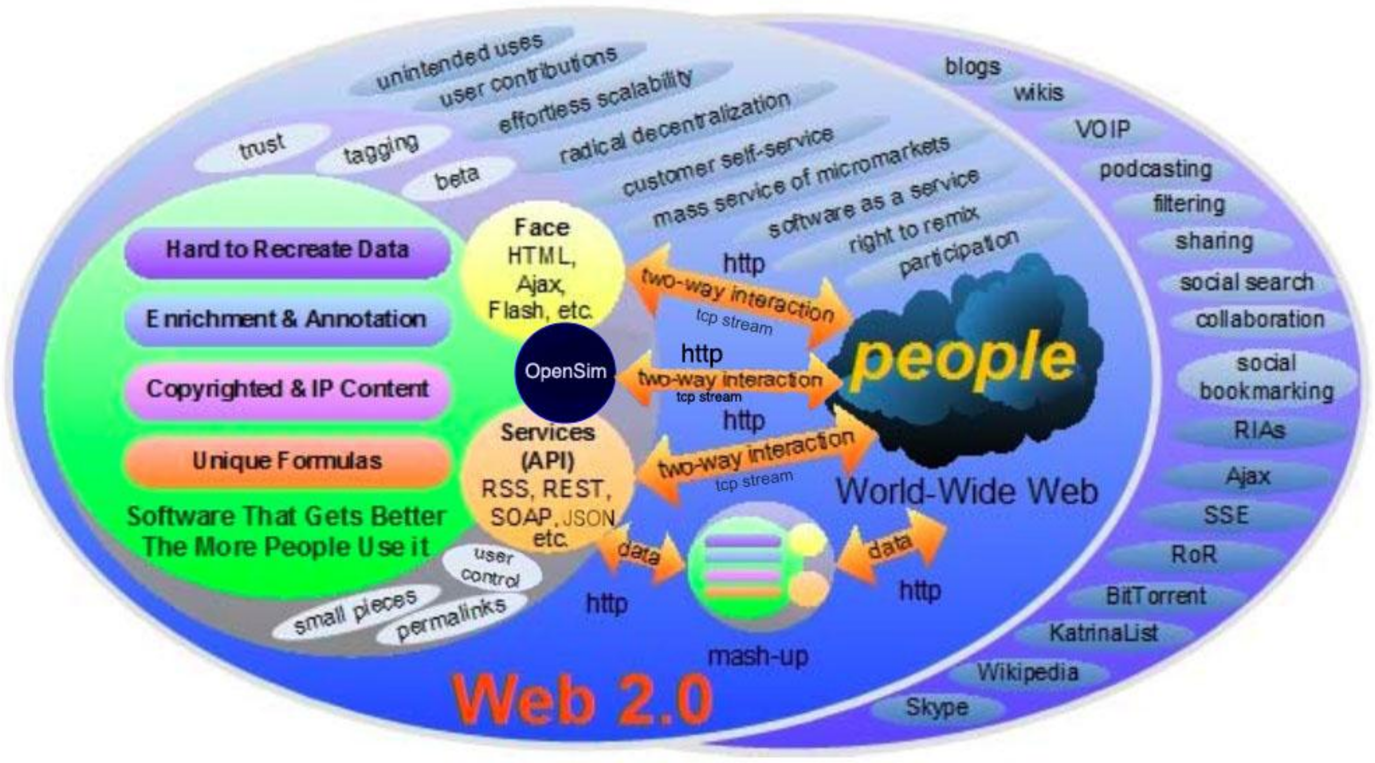
Exercises

- 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!


unintended uses
user contributions
effortless scalability
radical decentralization
customer self-service
mass service of micromarkets
software as a service
right to remix
participation
trust
tagging
beta
Hard to Recreate Data
Enrichment & Annotation
Copyrighted & IP Content
Unique Formulas
Software That Gets Better The More People Use it
Face HTML, Ajax, Flash, etc.
OpenSim
Services (API) RSS, REST, SOAP, JSON etc.
http
two-way interaction
tcp stream
people
World-Wide Web
data
mash-up
user control
small pieces
permalinks
Web 2.0
blogs
wikis
VOIP
podcasting
filtering
sharing
social search
collaboration
social bookmarking
RIAs
Ajax
SSE
RoR
BitTorrent
KatrinaList
Wikipedia
Skype