CSS: the language of Web design

TI1506: Web and Database Technology
Claudia Hauff

Lecture 5 [Web], 2014/15

Course overview [Web]

1. http: the language of Web communication
2. HTML: the language of the Web
3. JavaScript: interactions in the browser
4. node.js: JavaScript on the server
5. **CSS: the language of Web design**
6. Ajax: asynchronous JavaScript
7. Personalization: Cookies & sessions
8. Securing your application

At the end of this lecture, you should be able to …

- **Position and style** HTML elements according to a given design of a Web page
- **Describe** the box model
- **Employ** pseudo-classes and pseudo-elements
- **Employ** CSS's data access/creation facilities and **reflect** upon them

A bit of context

A brief history of CSS

- **CSS 1**: a W3C recommendation in 1996
  - Support for fonts, colors, alignment, margins, ids and classes
- **CSS 2**: a W3C recommendation in 1998
  - Support added for media queries, element positioning
- **CSS 2.1**: a W3C recommendation in 2011
  - Fixed errors and added support for features widely implemented in major browsers
- **CSS 3**: currently under development; specification is split up into modules; progress varies between modules

  http://www.w3.org/Style/CSS/current-work
- **CSS 4**: some modules have reached “level 4” status

CSS 3+: a tale of many modules

Completed	Current	Upcoming
CSS Snapshot 2010	NOTE	
CSS Snapshot 2007	NOTE	
CSS Color Level 3	REC	REC
CSS Namespaces	REC	REC
Selectors Level 3	REC	REC
CSS Level 2 Revision 1	REC	REC
CSS Level 1	REC	
CSS Print Profile	NOTE	
Media Queries	REC	REC
CSS Style Attributes	REC	REC

Abbreviation	Full name
WD	Working Draft
LC	Last Call
CR	Candidate Recommendation
PR	Proposed Recommendation
REC	Recommendation

CSS 3+: a tale of many modules

Testing	Current	Upcoming
CSS Backgrounds and Borders Level 3	CR	PR
CSS Conditional Rules Level 3	CR	CR
CSS Image Values and Replaced Content Level 3	CR	PR
CSS Marquee	CR	PR
CSS Multi-column Layout	CR	CR
CSS Speech	CR	PR
CSS Values and Units Level 3	CR	PR
CSS Flexible Box Layout	LC	CR
CSS Text Decoration Module Level 3	CR	PR
CSS Cascading and Inheritance Level 3	CR	PR
CSS Fonts Level 3	CR	PR
CSS Writing Modes Level 3	CR	PR
CSS Shapes	CR	PR
CSS Masking	CR	PR
CSS Mobile Profile 2.0	CR	PR
CSS TV Profile 1.0	CR	?

Abbreviation	Full nar
WD	Working
LC	Last Call
CR	Candidat
PR	Proposed
REC	Recommendation

CSS 3+: a tale of many modules

Refining	Current	Upcoming
CSS Animations	WD	WD
CSS Counter Styles Level 3	LC	CR
CSS Text Level 3	LC	CR
CSS Fr		
CSS Tr		
CSS Tr		
Cascac		
Compc		
CSS Sy		

Revising	Current	Upcoming
CSS Box Alignment Module Level 3	WD	WD
CSS Grid Layout	WD	WD
CSS Paged Media Level 3	WD	LC
CSS Basic User Interface Level 3	CR	LC
CSSOM View	WD	WD
Selectors Level 4	WD	WD

Abbreviation	Full name
WD	Working Draft
LC	Last Call
CR	Candidate Recommendation
PR	Proposed Recommendation
REC	Recommendation

CSS 3

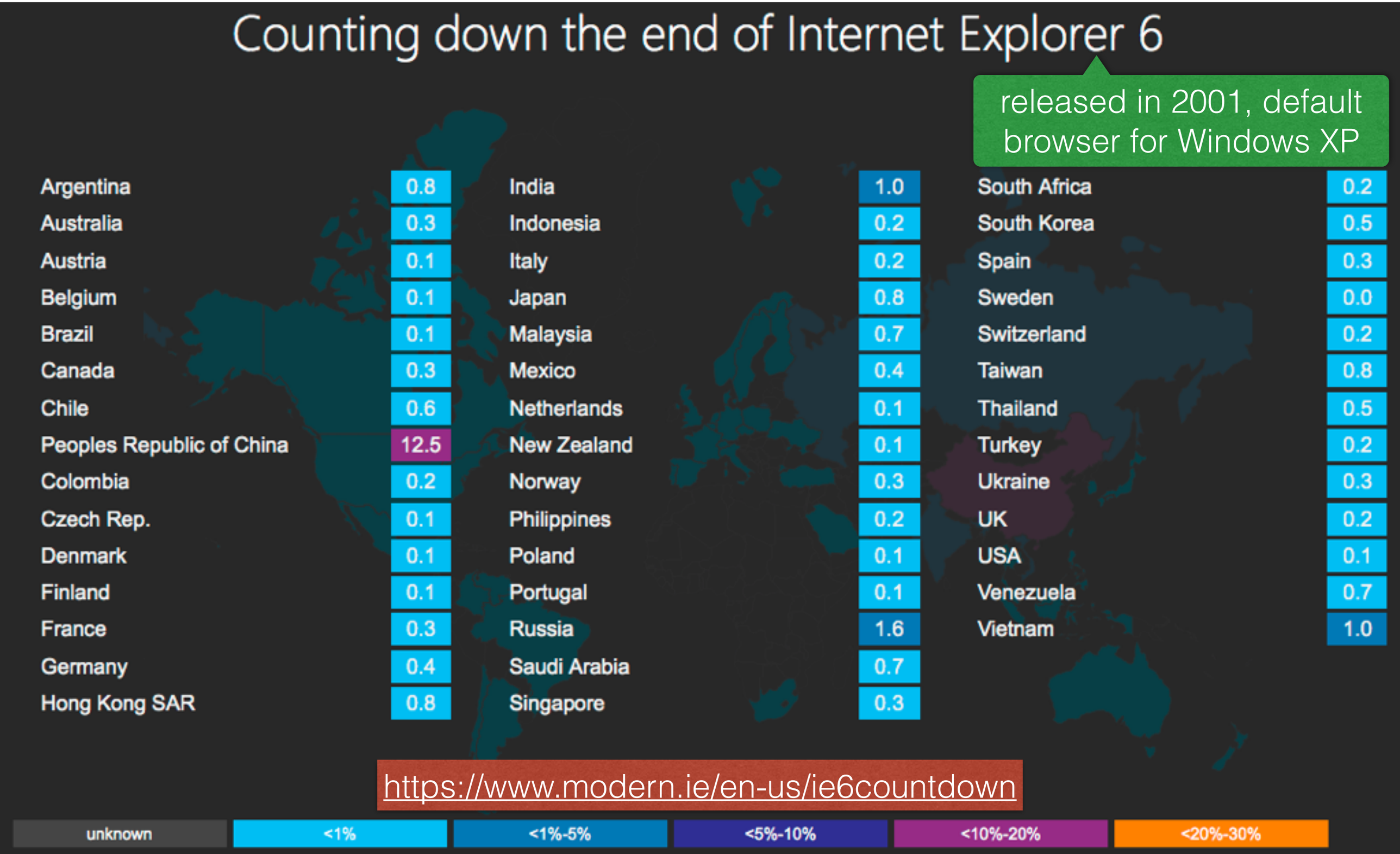
- Impossible to write CSS that relies on modern features and works across all browsers

- Implementation of CSS 3 features should be decided based on
  - **intended users** (mostly in the US or China or … ?),
  - the **mode of usage** (smartphone, touch-screen or desktop?),
  - the **type** of Web application (3D animations may not be necessary for a Todo list app)

- JavaScript libraries exist to help front-end developers to build cross-browser apps e.g. Modernizr

Requirement for our todo Web app: it should work on 3 **major modern** browsers

Are older browsers widespread?



Revision: chapter 3

Chapter 3 of the course book

CSS describes how elements in the DOM should be rendered.

```
1  body {
2    background-color: #ffee22;
3    width: 800px;
4    margin: auto;
5  }
6  h1 {
7    color: maroon;
8  }
9  p li {
10   color: gray;
11   border: 1px solid gray;
12 }
13 p.last {
14   color: green;
15 }
```

selector

property

value

CSS file

- Three types of style sheets:
  (1) browser's style sheet
  (2) author's style sheet
  (3) user's style sheet

  overrides
- Style sheets are processed in order; later declarations trump earlier ones
- **`!important`** overrides all other declarations (do not use if at all possible)

Pseudo-elements and pseudo-classes

Pseudo-class

Pseudo-class: a keyword added to a **selector** which indicates **a particular state** of the corresponding element.

Pseudo-classes (and pseudo-elements) allow styling according to **document external** factors (e.g. mouse movements, user browsing history).

```
1 selector:pseudo-class {
2   property: value;
3   property: value;
4 }
```

in general

Pseudo-class

- More than 30 pseudo-classes
- Support varies according to the **rendering engine**

> **A rendering engine** (or browser engine, layout engine) is responsible for translating HTML+CSS (among others) to the screen.

Rendering engine	Browser
Gecko	Firefox
Trident	Internet Explorer
WebKit	Safari, older version of Google Chrome
Blink	Google Chrome (new versions), Opera

Popular pseudo-classes

`:hover`	a pointing device (mouse) hovers over the element
`:active`	the element is currently being active (e.g. clicked)

```
button {
  background: white;
  color: darkgray;
  width:100px;
  padding:5px;
  font-weight:bold;
  text-align: center;
  border:1px solid darkgray;
}
```

```
button:hover {
  color:white;
  background:darkgray;
}

button:active {
  border:1px dashed;
  border-color: black;
}
```

http://jsfiddle.net/0g2eLcjf/

Popular pseudo-classes

`:enabled`	an element that can be clicked/selected
`:disabled`	an element that cannot be clicked/selected

```
1 button {
2    …
3 }
4
5 button:enabled:hover {
6    …
7 }
8
9 button:enabled:active {
10   …
11 }
```

HTML file

```
1 <button id="addyesterday"
2    disabled>
3       ADD TODO
4 </button>
```

- Enabled/disabled buttons look the same
- Enabled buttons change their look when being activated or at hovering

Popular pseudo-classes

`:nth-child(X)`	any element that is the Xth child element of its parent
`:nth-of-type(X)`	any element that is the Xth sibling of its type

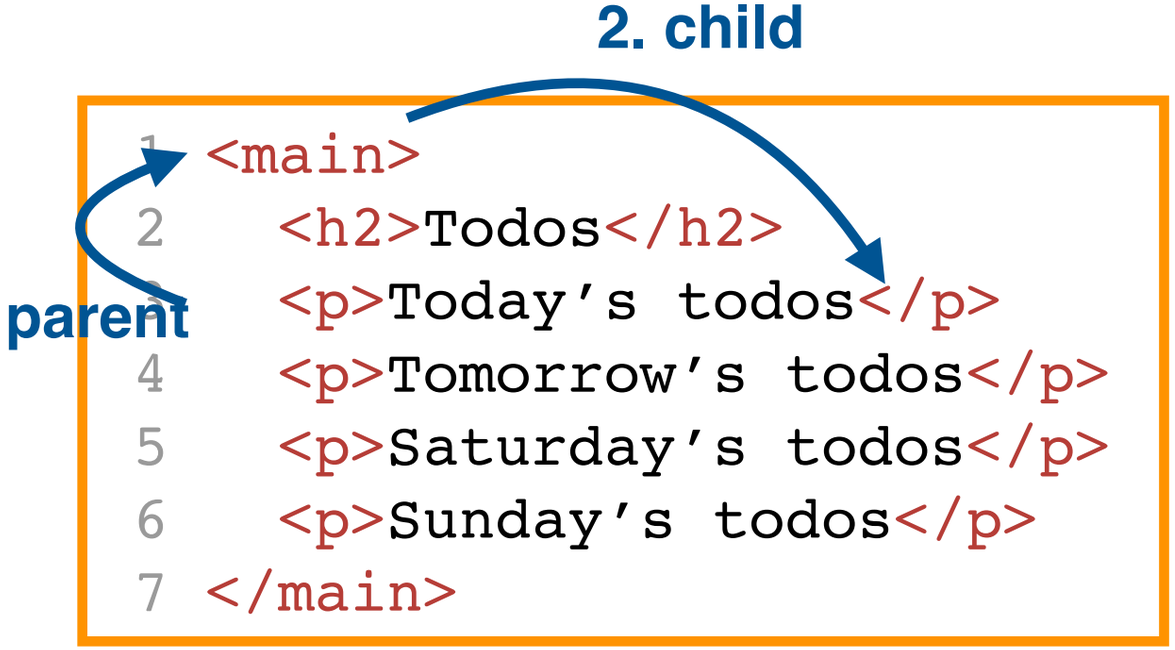
```
<main>
  <h2>Todos</h2>
  <p>Today's todos</p>
  <p>Tomorrow's todos</p>
  <p>Saturday's todos</p>
  <p>Sunday's todos</p>
</main>
```

```
p:nth-child(2) {
  color:red;
}

p:nth-of-type(2) {
  background-color:green;
}
```

Popular pseudo-classes

`:nth-child(X)`	any element that is the Xth child element of its parent
`:nth-of-type(X)`	any element that is the Xth sibling of its type




```
1 p:nth-child(2) {
2   color:red;
3 }
4
5 p:nth-of-type(2) {
6   background-color:green;
7 }
```

Popular pseudo-classes

`:nth-child(X)`	any element that is the Xth child element of its parent
`:nth-of-type(X)`	any element that is the Xth sibling of its type

sib-lings

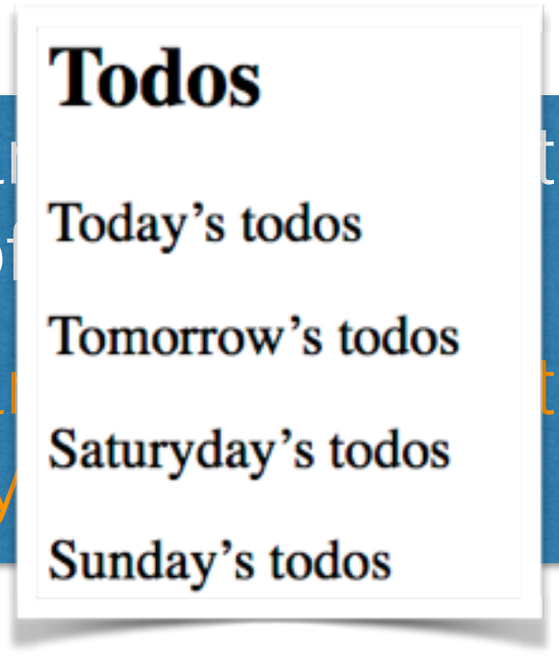
```
1 <main>
2   <h2>Todos</h2>
3   <p>Today's todos</p>
4   <p>Tomorrow's todos</p>
5   <p>Saturday's todos</p>
6   <p>Sunday's todos</p>
7 </main>
```

```
1 p:nth-child(2) {
2   color:red;
3 }
4
5 p:nth-of-type(2) {
6   background-color:green;
7 }
```

Popular pseudo-classes

`:nth-child(X)` a… t is …ent

`:nth-of-type(X)` a… t is …ts




```html
<main>
  <h2>Todos</h2>
  <p>Today's todos</p>
  <p>Tomorrow's todos</p>
  <p>Saturday's todos</p>
  <p>Sunday's todos</p>
</main>
```

```css
p:nth-child(2) {
  color:red;
}

p:nth-of-type(2) {
  background-color:green;
}
```

Popular pseudo-classes

`:nth-child(X)`	any element that is the Xth of its parent
`:nth-of-type(X)`	any element that is the Xth type


```html
<main>
  <h2>Todos</h2>
  <p>Today's todos</p>
  <p>Tomorrow's todos</p>
  <p>Saturday's todos</p>
  <p>Sunday's todos</p>
</main>
```

```css
p:nth-child(3) {
  color:red;
}

p:nth-of-type(4) {
  background-color:green;
}
```

Popular pseudo-classes

`:first-child` is equivalent to `:nth-child(1)`

`:last-child` is equivalent to `:nth-last-child(1)`

`:first-of-type` is equivalent to `:nth-of-type(1)`

`:last-of-type` is equivalent to `:nth-last-of-type(1)`

Popular pseudo-classes

`:not(X)`

matches all elements that are not represented by selector X

```
1 <main>
2    <h2>Todos</h2>
3    <p id=“firsttodo">Today's todos</p>
4    <p class=“todo">Tomorrow's todos</p>
5    <p class=“todo">Saturyday's todos</p>
6    <p>Sunday’s todos</p>
7 </main>
```

```
1 main :not(.todo) {
2      color:orange;
3 }
```

Popular pseudo-classes

`:in-range :out-of-range`

can be used to style elements with range limitations

```
1 <main>
2     <input type="text" placeholder="add your todo" />
3     <input id="dl" name="dl" type="number" min="1" max="30"
4         placeholder="Days to deadline" required />
5     <label for="deadline1"> </label>
6 </main>
```

```
1 input[type=text] {
2   border: 0px;
3   width: 150px;
4 }
5 input[type=number] {
6   width: 100px;
7 }
```

attribute selector

adjacent selector

```
1 input:valid + label::after {
2  content:" \2714";
3  color: rgba(0,100,0,0.7);
4 }
5
6 input:invalid + label::after {
7  content:" (invalid)";
8  color: rgba(255,0,0,0.7);
9 }
```

unicode

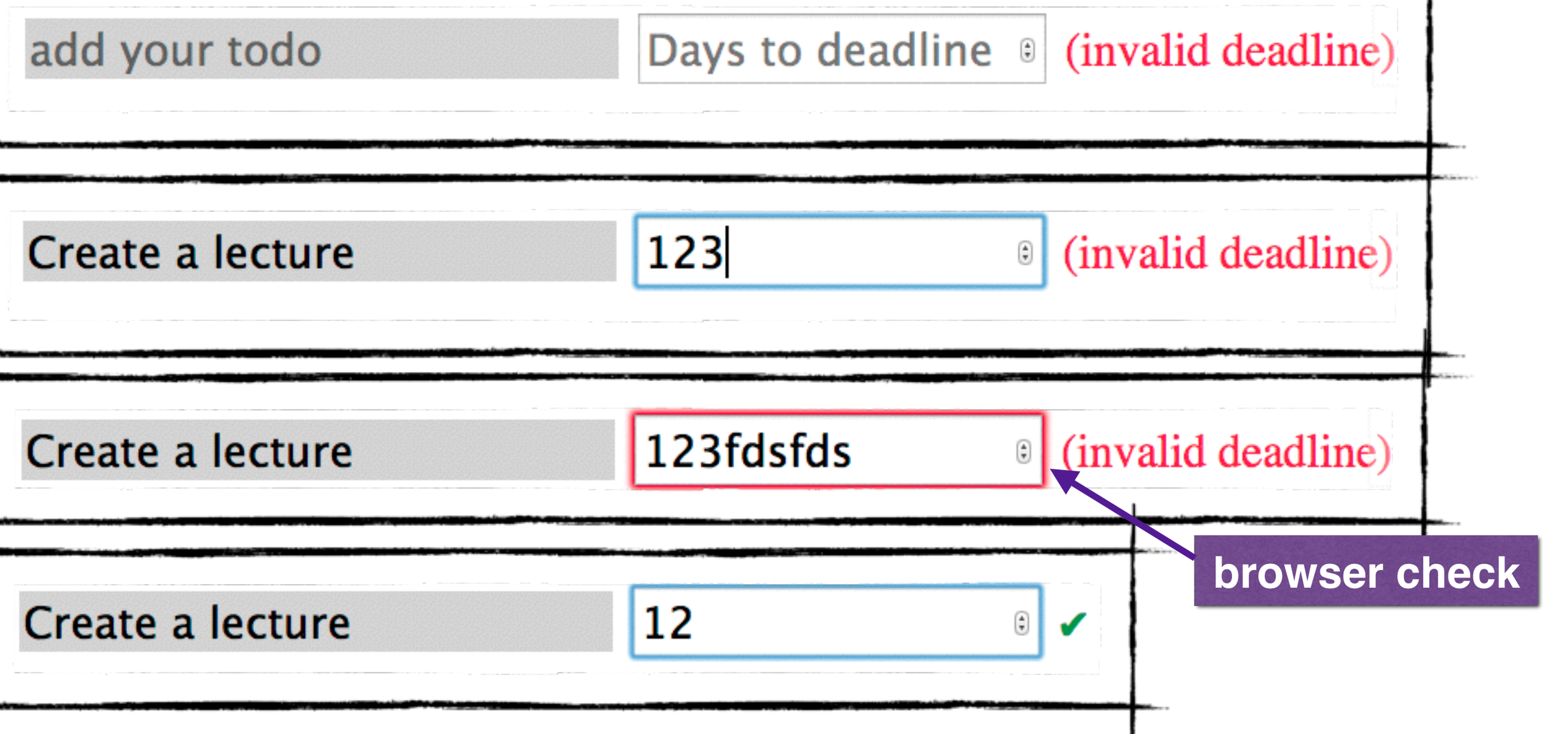
rgb & alpha

Popular pseudo-classes

http://jsfiddle.net/20p0qhy4/

`:in-range :out-of-range`

can be used to style elements with range limitations



Pseudo-elements

```
::first-letter
::first-line
```

Canonical example:
enlarge the first letter/line of a paragraph

```
1 p::first-line {
2     color:gray;
3     font-size:125%;
4 }
5
6 p::first-letter {
7     font-size:200%;
8 }
```

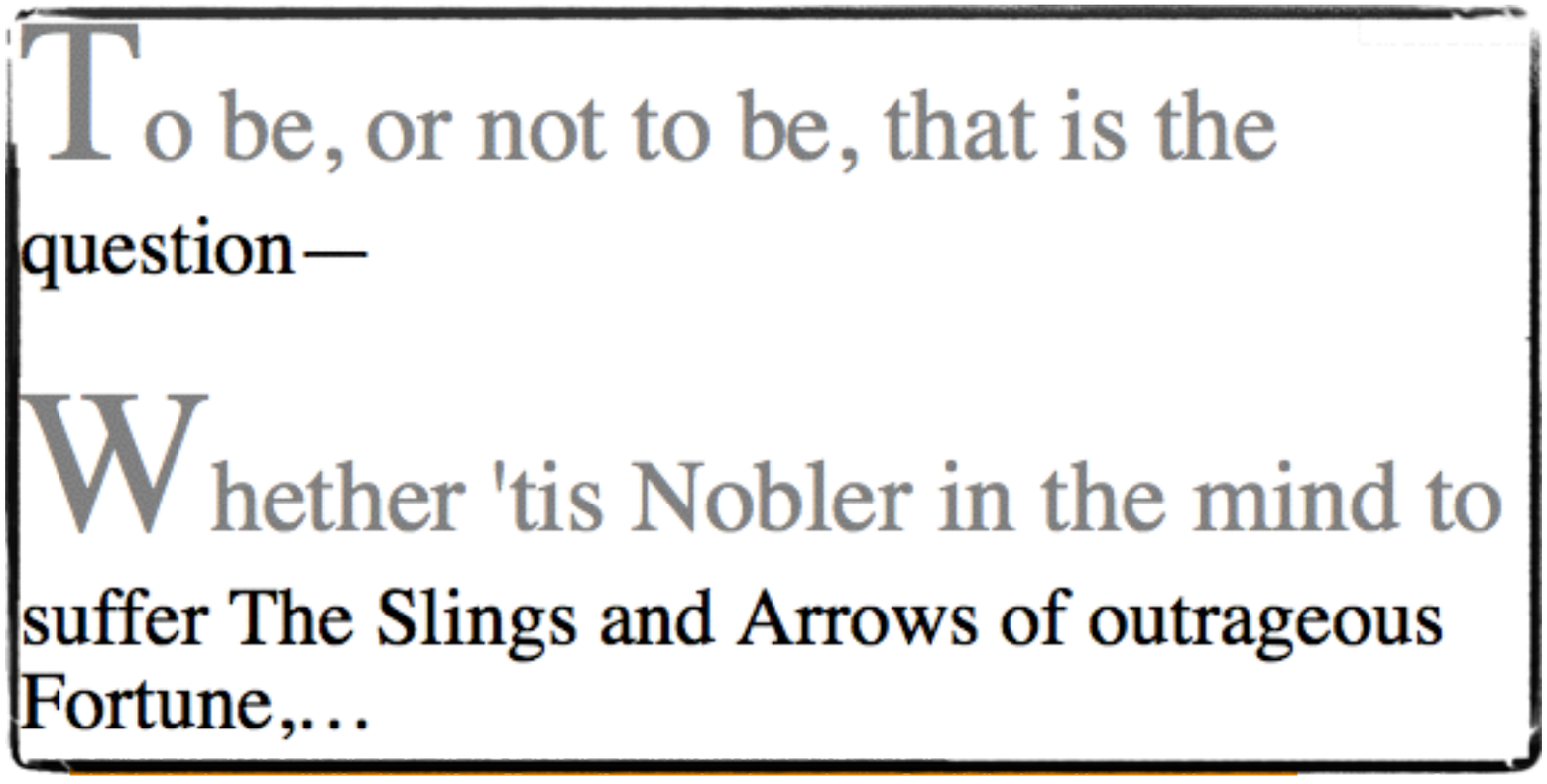
```
1  <p>
2    To be, or not to be, that
3    is the question—
4  </p>
5  <p>
6    Whether 'tis Nobler in the
7    mind to suffer
8    The Slings and Arrows of
9    outrageous Fortune,…
10 </p>
```

Pseudo-elements

```
::first-letter
::first-line
```

Canonical example:
enlarge the first letter/line of a paragraph

```
1 p::first-line {
2     color:gray;
3     font-size:125%;
4 }
5
6 p::first-letter {
7     font-size:200%;
8 }
```



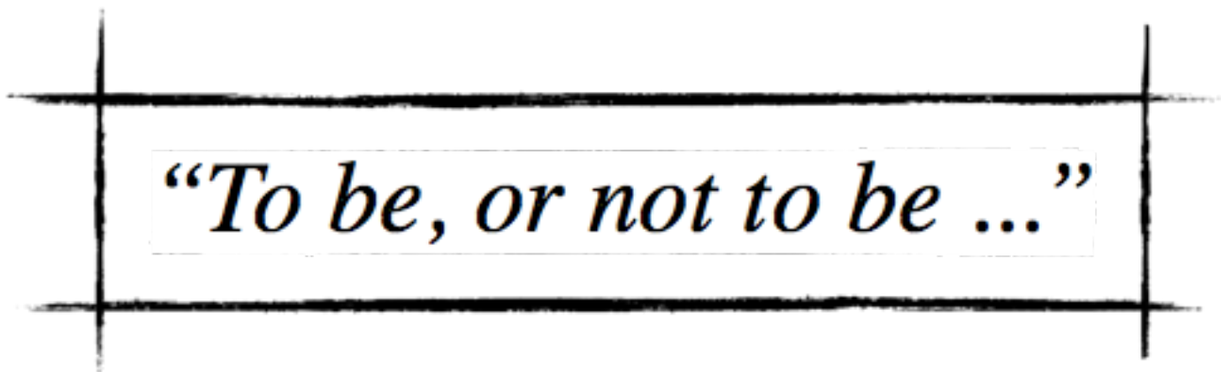
Pseudo-elements

`::after`	used to add (cosmetic) content after an element
`::before`	used to add (cosmetic) content before an element

```
1 <cite>
2   To be, or not
3   to be …
4 </cite>
```

```
1 cite::before {
2     content: "\201C";
3 }
4 cite::after {
5     content: "\201D";
6 }
```

Canonical example:
add quotation marks to quotes



Pseudo-elements

`::after`	used to add (cosmetic) content after an element
`::before`	used to add (cosmetic) content before an element

This also works, i.e. it leads to exactly the same output on screen.

```
1 <cite>
2 </cite>
```

```
1 cite::before {
2     content: "\201CTo be, or ";
3 }
4 cite::after {
5     content: "not to be ... \201D";
6 }
```

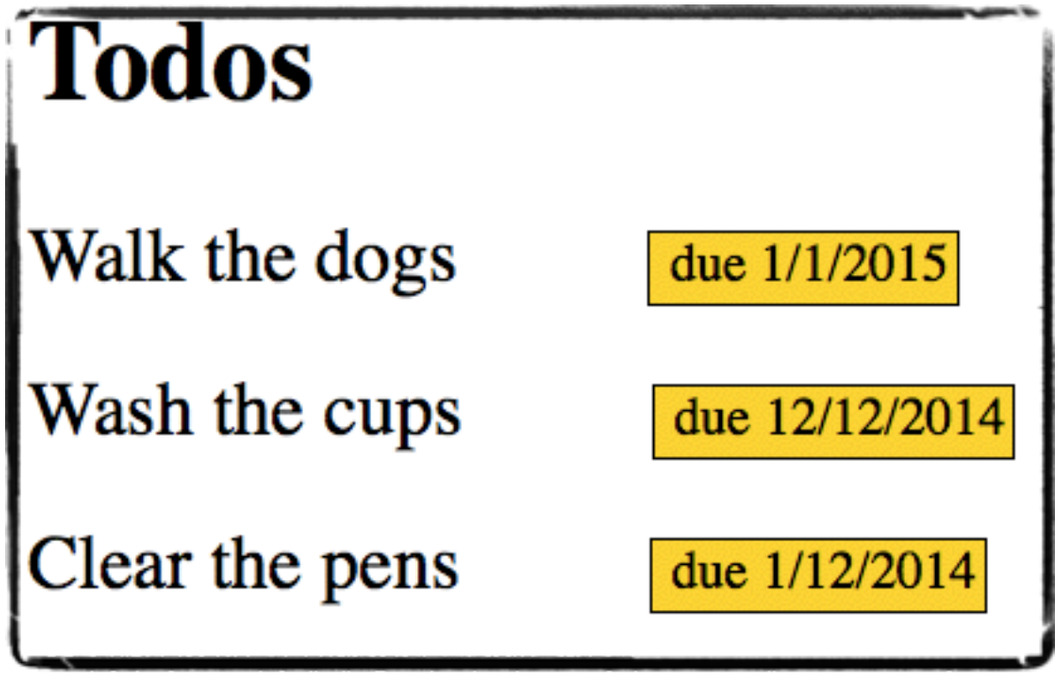
CSS can work with data!?

Data in CSS

CSS & data (one way)

CSS does not only describe the style, it can carry data too.

```
1 <main>
2    <h2>Todos</h2>
3     <p id="t1">Walk the dogs</p>
4     <p id="t2">Wash the cups</p>
5     <p id="t3">Clear the pens</p>
6 </main>
```




```
1  p::after {
2     background-color:gold;
3     border: 1px solid;
4     font-size: 70%;
5     padding: 2px;
6     margin-left: 50px;
7  }
8
9  p#t1::after {
10    content: " due 1/1/2015";
11 }
12
13 p#t2::after {
14    content: " due 12/12/2014";
15 }
16
17 p#t3::after {
18    content: " due 1/12/2014";
19 }
```

CSS & data (one way)

CSS does not only describe the style, it can carry data too.

```html
<main>
    <h2>Todos</h2>
     <p id="t1">Walk the dogs</p>
     <p id="t2">Wash the cups</p>
     <p id="t3">Clear the pens</p>
</main>
```

```css
p::after {
    background-color:gold;
    border: 1px solid;
    font-size: 70%;
    padding: 2px;
    margin-left: 50px;
}

p#t

}
```

"Inspect element" on http://jsfiddle.net/20p0qhy4/ (invalid is not in the DOM)

Issues:

1. Data is distributed across HTML and CSS files.
2. CSS is conventionally not used to store data.
3. Content is not part of the DOM (**accessibility problem!**)

CSS & data-* (the preferred way)

CSS makes use of data stored in HTML elements.

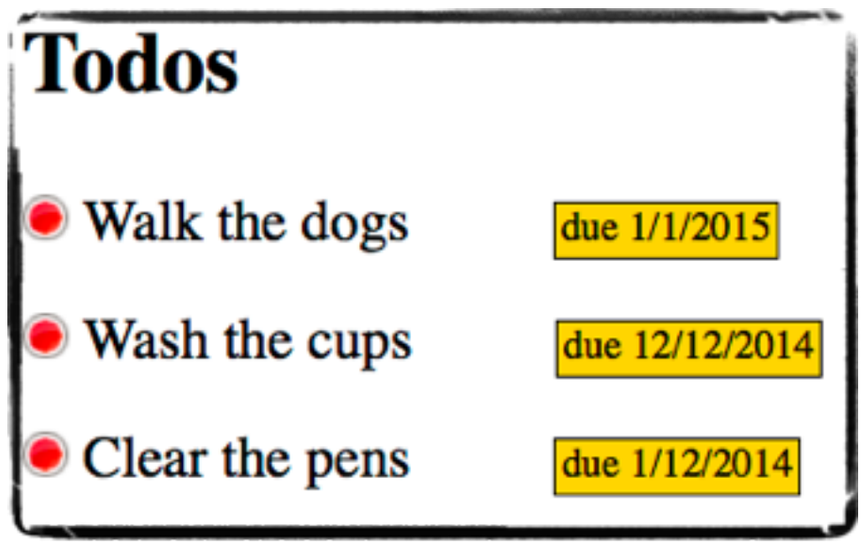
Recall: HTML elements can have data-* attributes.

```
1 <main>
2   <h2>Todos</h2>
3     <p id="t1" data-due="1/1/2015" >Walk the dogs</p>
4     <p id="t2" data-due="12/12/2014">Wash the cups</p>
5     <p id="t3" data-due="1/12/2014">Clear the pens</p>
6 </main>
```

```
1 p::after {
2    background-color:gold;
3    border: 1px solid;
4    font-size: 70%;
5    padding: 2px;
6    margin-left: 50px;
7    content: "due " attr(data-due);
8 }
9 p::before {
10   content: url(http://www.abc.de/dot.png);
11 }
```

attr() retrieves the value of an attribute

content attribute can also reference a url



CSS & data-* (the preferred way)

Another example: a simple tooltip

```
1 <ul>
2  <li data-name="Cascading Style Sheets">CSS</li>
3  <li data-name="HyperText Markup Language">HTML</li>
4  <li data-name="Hypertext Transfer Protocol">http</li>
5  <li data-name="Hypertext Transfer Protocol Secure">https</li>
6 </ul>
```

```
1  li {
2      cursor:help;
3  }
4  li:hover::after {
5     background-color:rgba(10,10,10,0.7);
6     color: gold;
7     border: 1px dashed;
8     padding: 5px;
9     font-size: 70%;
10    content: attr(data-name);
11    position:relative;
12    bottom:15px;
13    left:5px;
14 }
```

we can change the cursor type

- CSS
- HTML
- http
- https

Hypertext Transfer Protocol

CSS counters

CSS counters can count the number of times a ruleset is called. Counters are set and maintained by CSS.

```
1 <main>
2    <h2>Todos</h2>
3      <p id="t1" data-due="1/1/2015"  >Walk the dogs</p>
4      <p id="t2" data-due="12/12/2014">Wash the cups</p>
5      <p id="t3" data-due="1/12/2014">Clear the pens</p>
6 </main>
```

```
 1 body {
 2     /* initialize counter to 0 */
 3     counter-reset: countTodo;
 4 }
 5 p::before {
 6     /* increment at each <p> */
 7     counter-increment: countTodo;
 8     /* counter written out */
 9     content:" Todo " counter(countTodo)": ";
10 }
```

Todos

Todo 1: Walk the dogs

Todo 2: Wash the cups

Todo 3: Clear the pens

Nested CSS counters

Child elements receive their own counter instance.

Different counter instances are combined via **`counters()`.**

```
<ul>
 <li>Today's todos
   <ul>
     <li>Walk the dogs</li>
     <li>Wash the cups</li>
     <li>Clear the pens</li>
   </ul>
 </li>
 <li>Tomorrow's todos
   <ul>
     <li>Walk the dogs</li>
     <li>Wash the dishes</li>
   </ul>
 </li>
</ul>
```

```
ul {
  counter-reset: cli;
  list-style-type: none;
}

li::before {
  counter-increment: cli;
  content: counters(cli,".") ": ";
}
```

1: Today's todos
    1.1: Walk the dogs
    1.2: Wash the cups
    1.3: Clear the pens
2: Tomorrow's todos
    2.1: Walk the dogs
    2.2: Wash the dishes

Deciding which CSS features to use

Can I use `attr()`?

Is it an established (accepted) part of the CSS specification?

1. **W3C CSS specification**
   - **Candidate Recommendation or Recommendation?**
   - **CSS2 or CSS3?**
   - **Exhaustive overview of all aspects (by necessity)**
2. **Mozilla Developer Network**
   - **Focuses on the most important aspects of a technology (not exhaustive)**
   - **Up-to-date information**
   - **Easy to get a quick overview**

Can I use `attr()`?

Specifications

Specification	Status	Comment
CSS Values and Units Module Level 3 The definition of 'attr()' in that specification.	CR Candidate Recommendation	Added two optional parameters; can be used on all properties; may return other values than `<string>`. These changes are ⚗ and may be dropped during the CR phase if browser support is too small.
CSS Level 2 (Revision 1) The definition of 'attr()' in that specification.	REC Recommendation	Limited to the `content` property; always return a `<string>`}

Great! Not just for `content`.

Experimental! Use at own risk.

okay to use for `content`

strings only

Browser compatibility

Mobile browsers support `attr()` in `content`

Desktop | Mobile

Feature	Android	Firefox Mobile (Gecko)	IE Mobile	Opera Mobile	Safari Mobile
Basic support	2.1	1.0 (1.0)	8	10.0	3.1
Usage in other properties than `content`and with non-string values ⚗	?	Not supported (see bug 435426)	Not supported	?	?

https://developer.mozilla.org/en-US/docs/Web/CSS/attr

Browser-specific prefixes

CSS is under active development, many features are not stable,

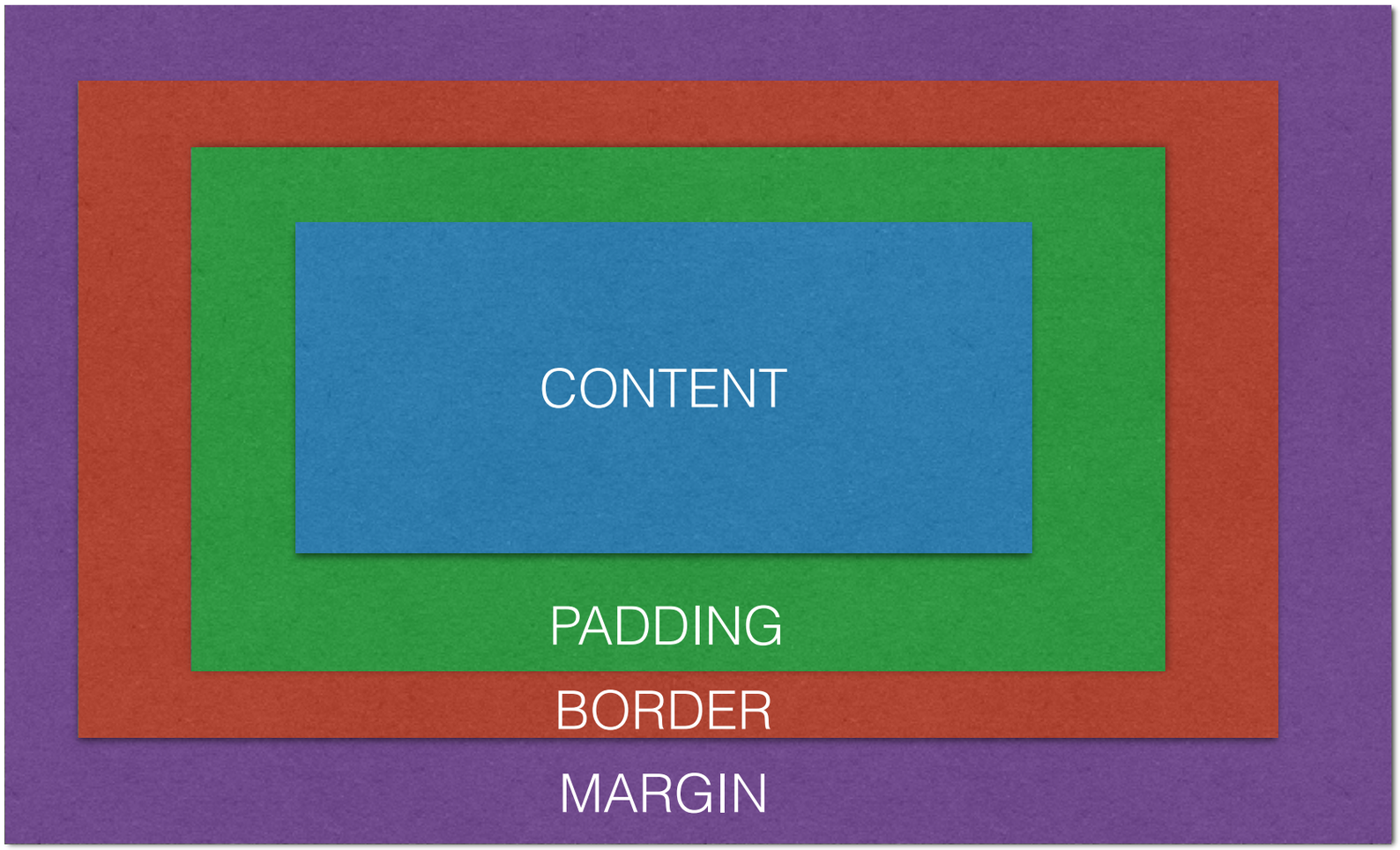
- are often used with browser vendor prefixes, and,
- might change in the future (as the specification changes).

```
1 main:-webkit-full-screen {
2 } /* Chrome */
3
4 main:-moz-full-screen {
5 } /* Firefox */
6
7 main:-ms-fullscreen {
8 } /* Internet Explorer */
9
10 main:fullscreen {
11 } /* W3C proposal */
```

- **Advantage**: exciting new features can be used early on
- **Disadvantage**: a new browser release might break the implemented CSS

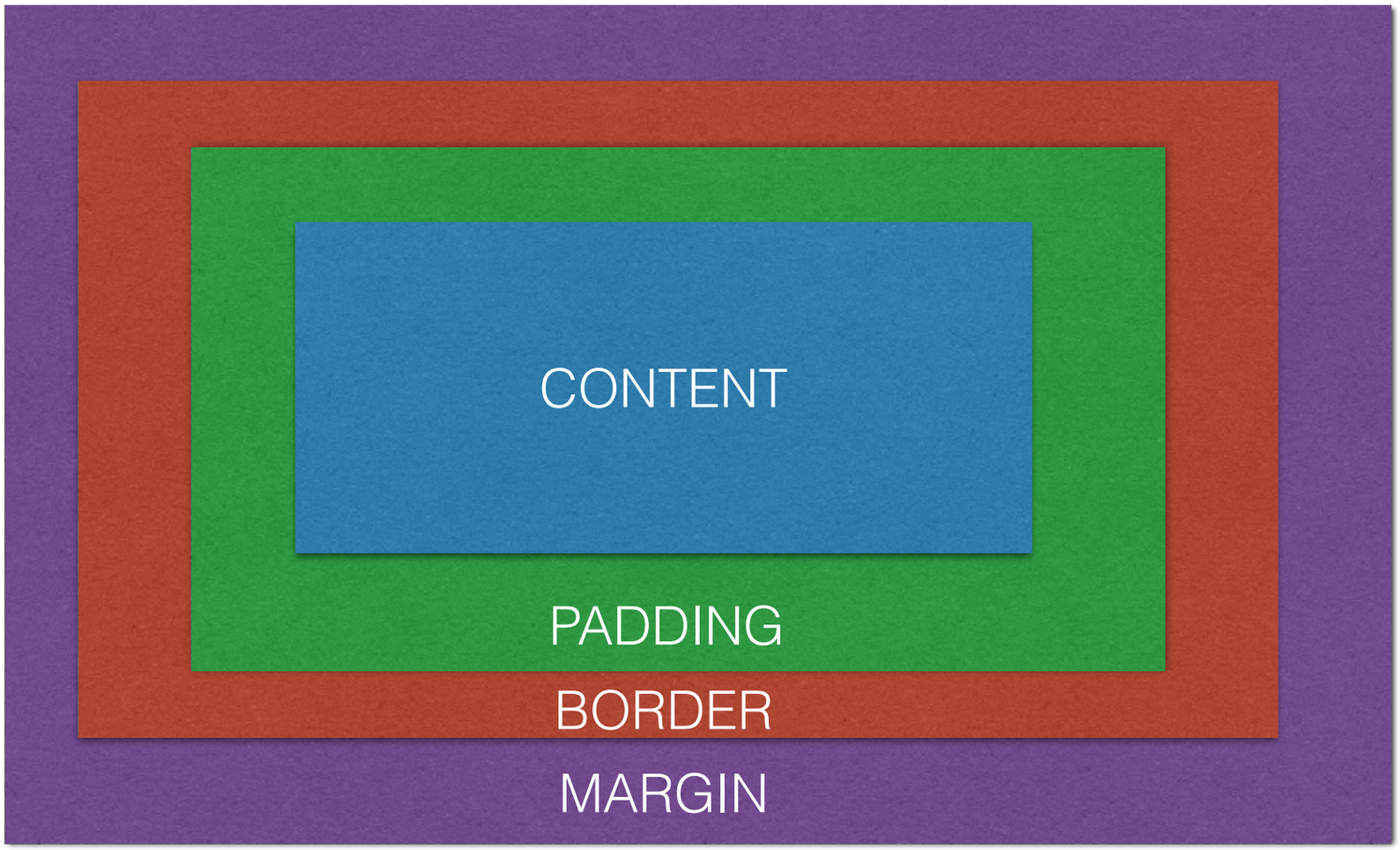
The box model

The box model



- **Padding**: empty area between content and border
- **Border**: area of the border
- **Margin**: empty area between this element and its neighbours

The box model

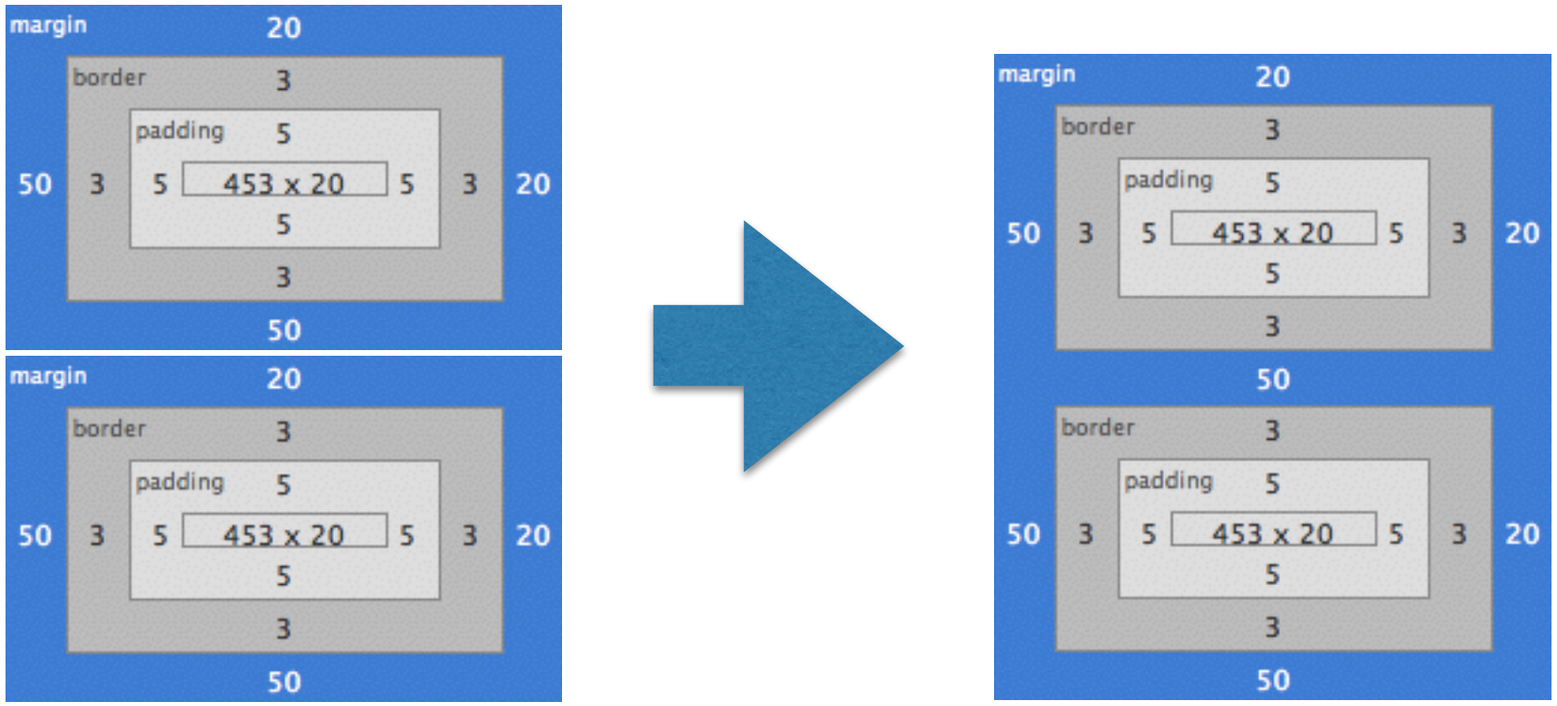


Demo: look at an example Web page with Firebug's Inspect Element.

- on the fly editing to see the effects possible (including adding new properties)
- e.g. change size of "Doorzoek de site"

Margins

- Margins can **collapse**: top/bottom margins of blocks collapse into one margin (the largest one)



Element positioning with `float`, `position` and `display`

Elements “flow” by default

Block-level are surrounded by line-breaks. They can contain block-level and inline elements. The width is determined by their containing element.

e.g. `<main>` or `<p>`

HTML 4 terminology

Inline elements can be placed within block/inline elements. They can contain other inline elements. The width is determined by their content.

e.g. `<span>` or `<a>`

```
<main>
<p>
  This is a paragraph containing <a href="#">a link</a>
</p>
<p>
  This is another paragraph
  <span>
    with a span and <a href="#">a link in the span</a>
  </span>
</p>
</main>
```

Elements “flow” by default

Block-level are surrounded by line-breaks. They can contain block-level and inline elements. The width is determined by their containing element.

e.g. `<main>` or `<p>`

HTML 4 terminology

Inline elements can be placed within block/inline elements. They can contain other inline elements. The width is determined by their content.

e.g. `<span>` or `<a>`

This is a paragraph containing a link

This is another paragraph with a span and a link in the span

```
1 main {width: auto;}
```

Elements "flow" by default

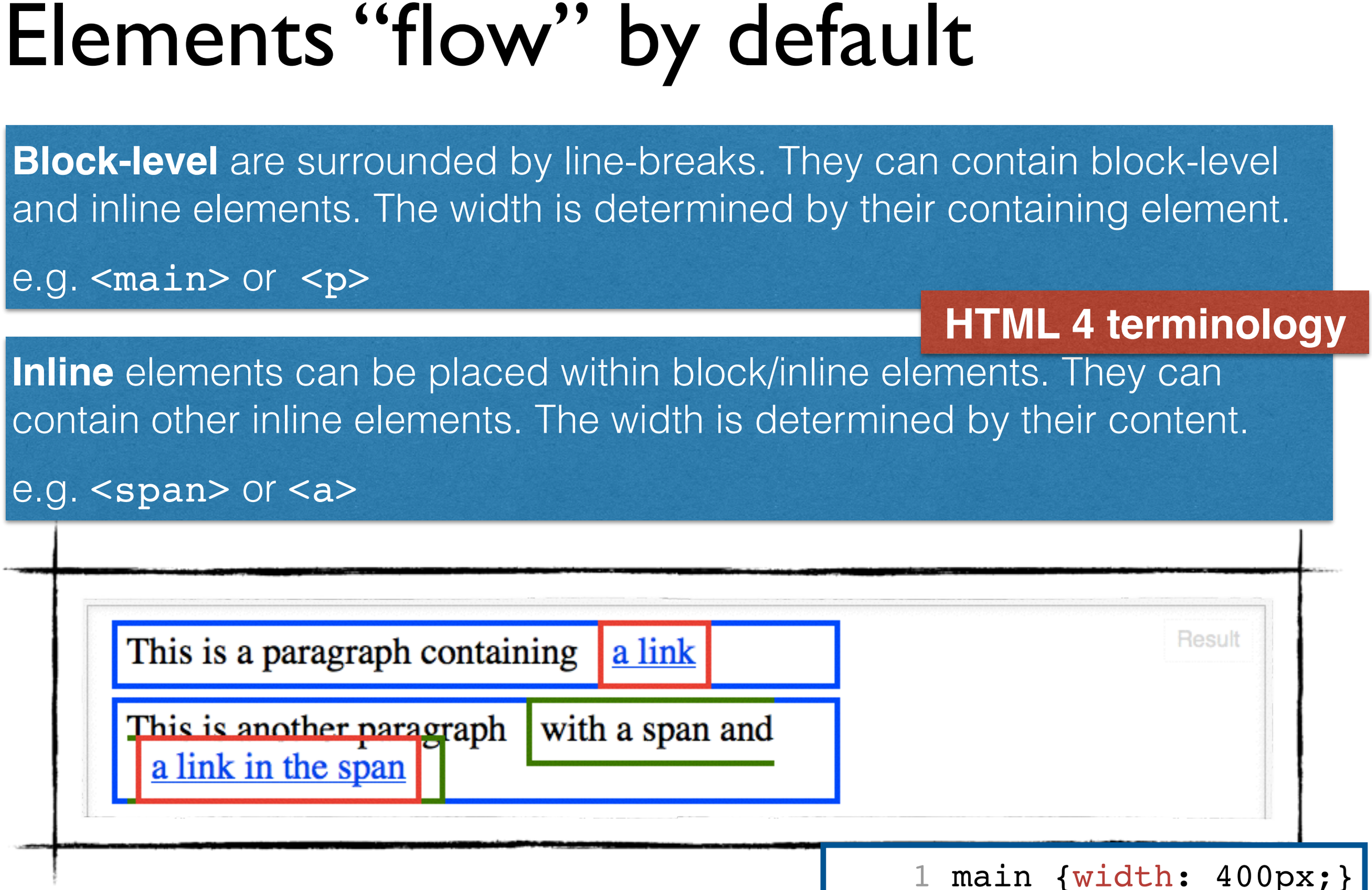
Block-level are surrounded by line-breaks. They can contain block-level and inline elements. The width is determined by their containing element.

e.g. `<main>` or `<p>`

HTML 4 terminology

Inline elements can be placed within block/inline elements. They can contain other inline elements. The width is determined by their content.

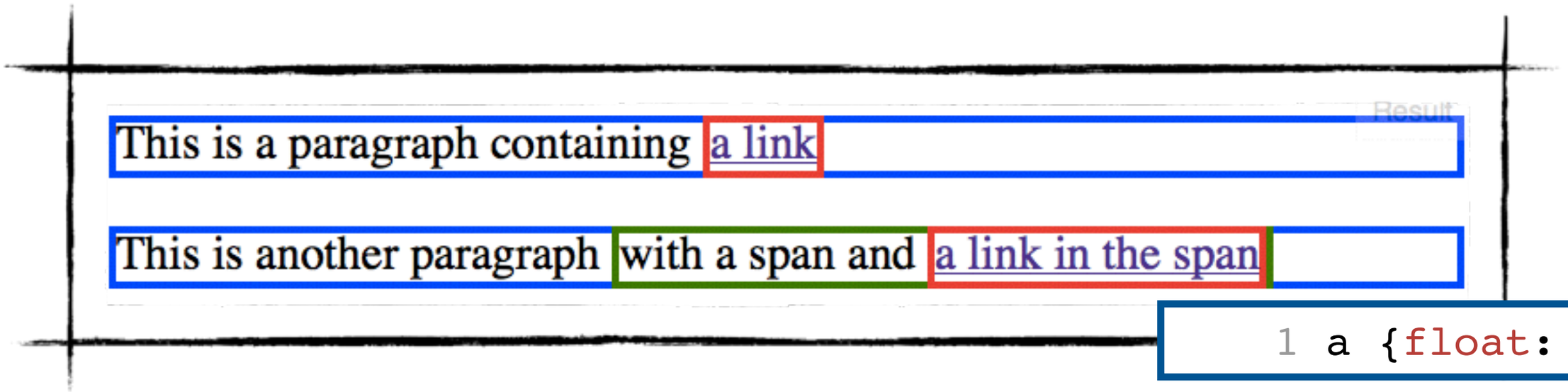
e.g. `<span>` or `<a>`




```
1 main {width: 400px;}
```

Taking elements out of the flow

`float:left` (or **`:right`**) takes an element out of the flow; it is moved to the leftmost (or rightmost) possible position in the containing element —either the element edge or another float.




```
1 a {float: none;}
```

Taking elements out of the flow

`float:left` (or **`:right`**) takes an element out of the flow; it is moved to the leftmost (or rightmost) possible position in the containing element —either the element edge or another float.

Result

a link This is a paragraph containing

a link in the span This is another paragraph with a span and

```
1 a {float: left;}
```

Result

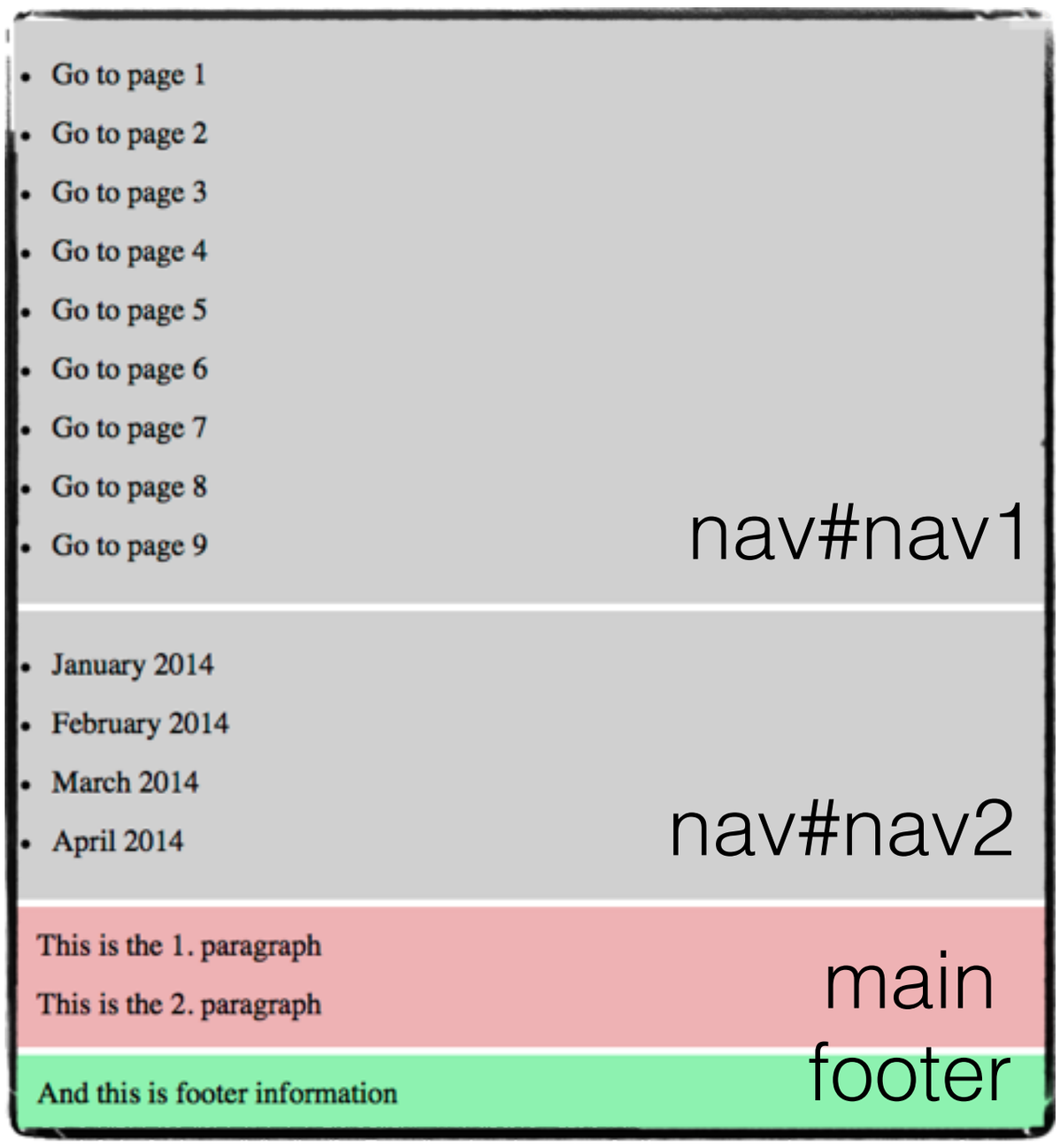
This is a paragraph containing a link

This is another paragraph with a span and a link in the span

```
1 a {float: right;}
```

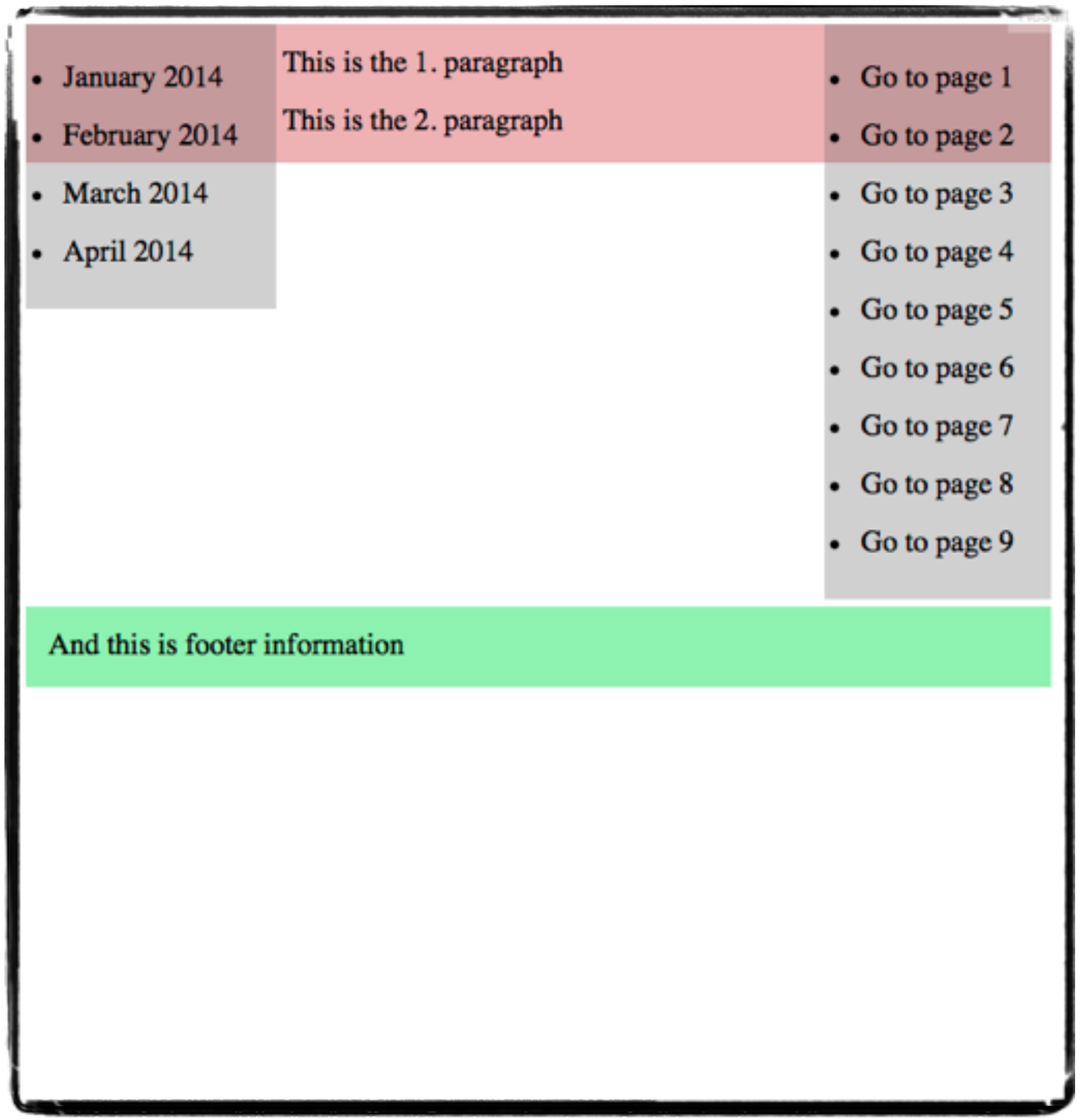
Resetting the flow with `clear`

canonical example: adding sidebars to a layout



Resetting the flow with `clear`

canonical example: adding sidebars to a layout




```
1 #nav1 {float: right;}
2 #nav2 {float: left;}
3 footer{clear: left;}
4 footer{clear: right;}
```

```
3 footer{clear: both;}
```

can be used instead

Fine-grained movement of elements: `position`

`position` enables elements to be moved around in any direction (up/down/left/right) by absolute or relative units.

`position:static`	the default
`position:relative`	the element is adjusted on the fly, other elements are not affected
`position:absolute`	element is taken out of the normal flow (no space is reserved for it)
`position:fixed`	similar to absolute, but fixed to the viewport
`position:sticky`	in-between relative and fixed

CSS coordinate system

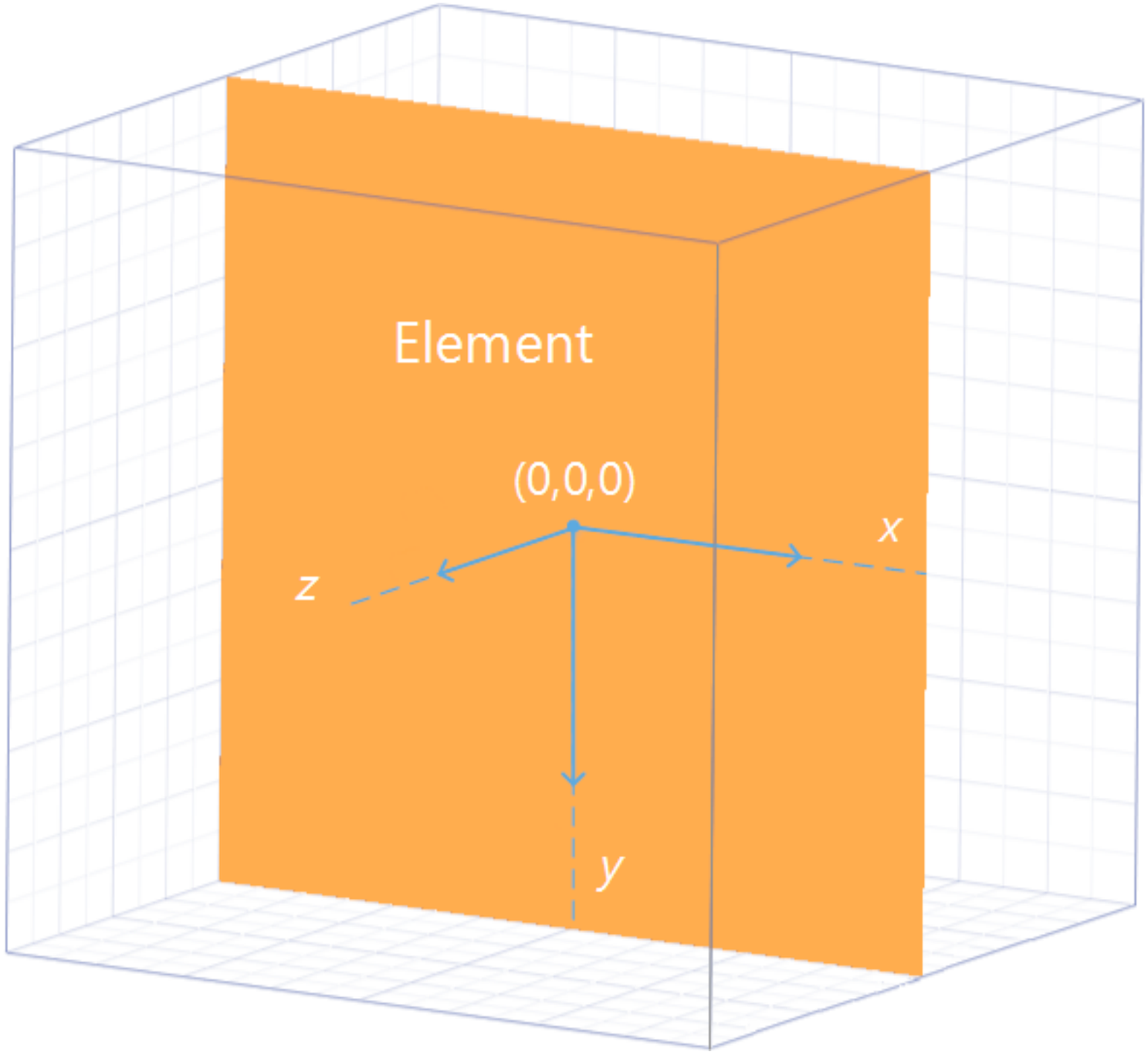


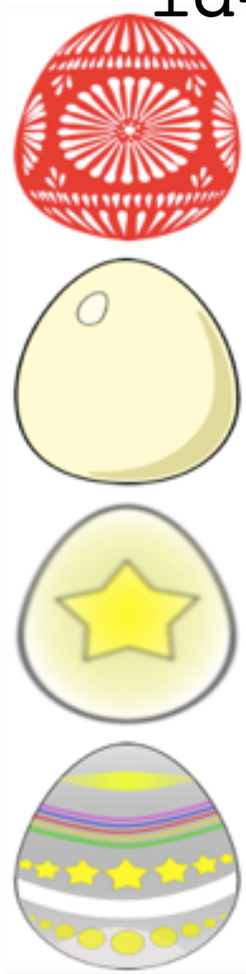
Image source: http://msdn.microsoft.com/en-us/library/ie/jj665791%28v=vs.85%29.aspx

position:relative

the element is adjusted on the fly, other elements are **not** affected

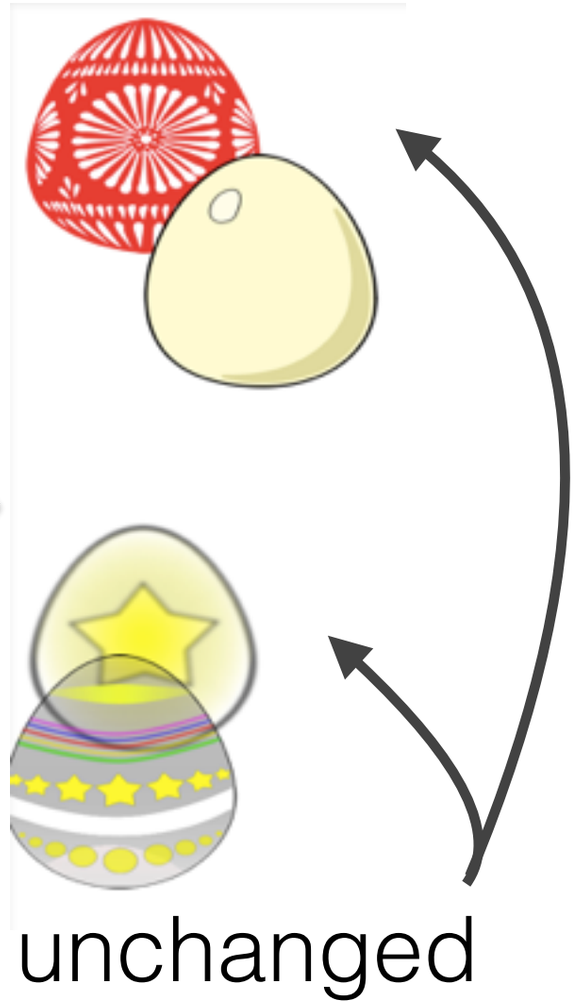
movement is **relative** to its original position

id="egg1"



id="egg4"

```
1  #egg2 {
2    position:relative;
3    bottom:20px;
4    left:20px;
5  }
6
7  #egg4 {
8    position:relative;
9    bottom:50px;
10    right:10px;
11 }
```



unchanged

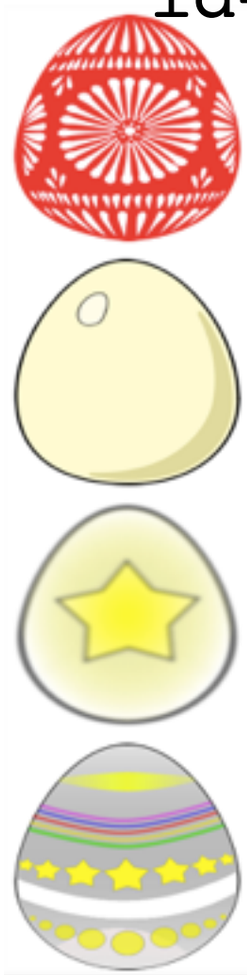
http://jsfiddle.net/zj800Lso/

position:absolute

the element is taken out of the normal flow (no space is reserved)

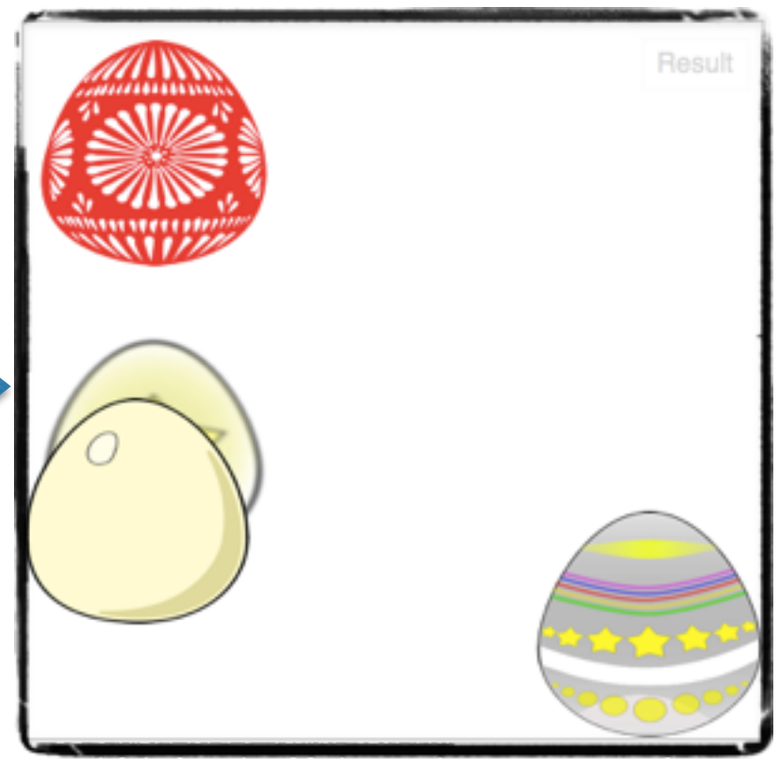
positioning is relative to nearest ancestor or the window

id="egg1"



id="egg4"

```
1  #egg2 {
2    position:absolute;
3    bottom:50px;
4    left:0px;
5  }
6
7  #egg4 {
8    position:absolute;
9    bottom:0px;
10    right:0px;
11 }
```



window

http://jsfiddle.net/zj800Lso/1/

`position:fixed`

similar to **`absolute`**, but the containing "element" is the viewport

area of the document visible in the browser

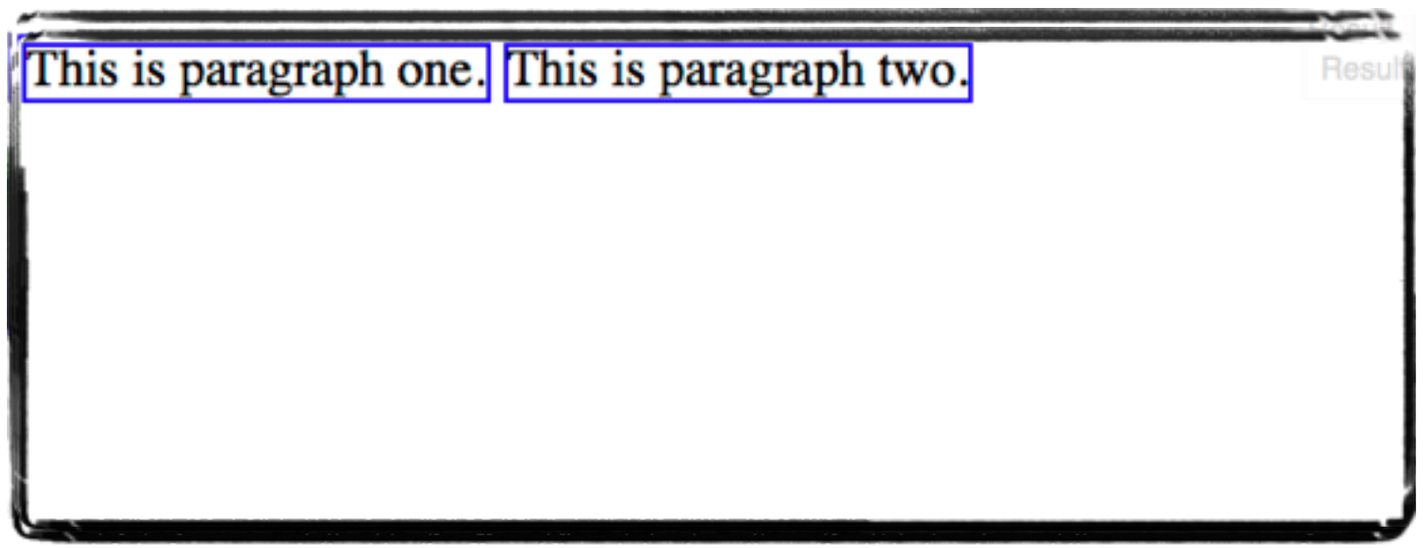
elements with **`position:fixed`** are always visible

http://jsfiddle.net/0g2eLcjf/2/

display

`display:inline`	element rendered with an inline element box
`display:block`	element rendered with a block element box
`display:none`	element (and its descendants) are hidden from view; no space is reserved in the layout

most useful to us




```
1 span {display: block; }
2 p    {display: inline;}
3 span {display: none;}
```

Today we covered

- the basics of CSS positioning
- the CSS box model
- CSS pseudo-classes and pseudo-elements

Readings

- **Required reading:** Chapter 3 of the Web course book

- Recommended: *The Book of CSS3: A Developer's Guide to the Future of Web Design*, Chapters 1-4 and 13