CSS: the language of Web design

Claudia Hauff
TI1506: Web and Database Technology
ti1506-ewi@tudelft.nl

TU Delft

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
- **Employ** pseudo-classes and pseudo-elements
- **Employ** CSS's data access/creation facilities and **reflect** upon them
- **Write** CSS media queries
- **Create** simple CSS-based animations

A bit of context

A brief history of CSS

A brief history of CSS

- **CSS 1**: a W3C recommendation in **1996**
  - Support for fonts, colors, alignment, margins, ids and classes

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

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

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

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

Abbreviation	Full name
WD	Working Draft
LC	Last Call
CR	Candidate Recommendation
PR	Proposed Recommendation
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 Fragmentation Level 3	WD	WD
CSS Transforms	WD	WD
CSS Transitions	WD	LC
Cascading Variables	WD	LC
Compositing and Blending	LC	CR
CSS Syntax Level 3	CR	PR

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 …?)
  - the **type** of Web app (are 3D animations necessary?)
- JavaScript libraries can help front-end developers to build cross-browser apps (e.g. `Modernizr`)

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

my_file.css

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

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

my_file.css

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

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

overrides

my_file.css

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

my_file.css

- 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** (if they are on the same level)

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

my_file.css

- 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** (if they are on the same level)
- **`!important`** overrides all other declarations

Pseudo-elements and pseudo-classes

A detour: the rendering engine

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

A detour: the rendering engine

- 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.

A detour: the rendering engine

- 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
from Trident to EdgeHTML	Internet Explorer
WebKit	Safari, older version of Google Chrome
Blink	Google Chrome (new versions), Opera

Pseudo-class

Pseudo-class

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

Pseudo-class

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

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

Pseudo-class

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

Pseudo-classes 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 }
```

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

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 <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>
```

Popular pseudo-classes

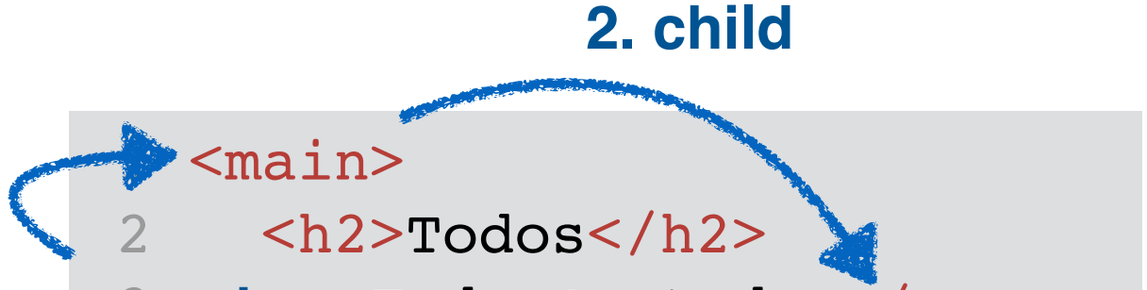
`:nth-child(X)`	any element that is the Xth child element of its parent
`:nth-of-type(X)`	any element that is the Xth sibling of its type

```
<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 <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)`	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**

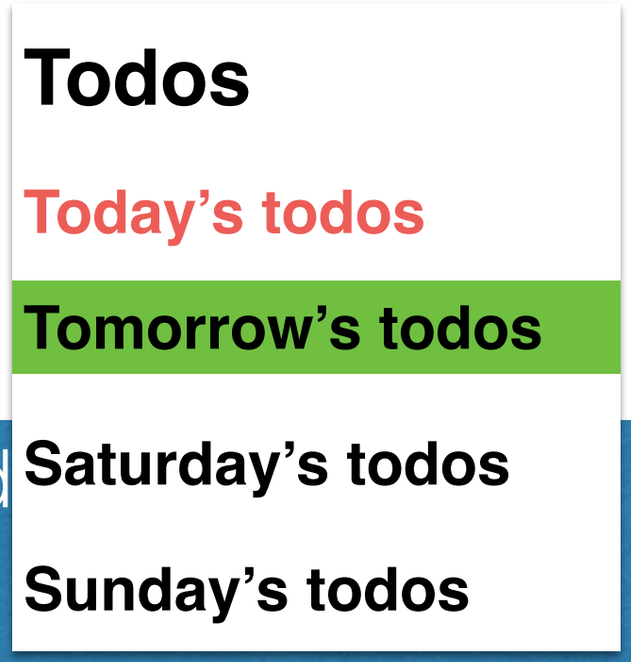
siblings

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



siblings

```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;
}
```

flickr@25363236@N07

What is the result of applying the CSS?

```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;
}
```

A

Todos

Today's todos

Tomorrow's todos

Saturday's todos

Sunday's todos

B

Todos

Today's todos

Tomorrow's todos

Saturday's todos

Sunday's todos

C

Todos

Today's todos

Tomorrow's todos

Saturday's todos

Sunday's todos

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

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

Popular pseudo-classes

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

```
1 button {
2   background: white;
3   color: darkgray;
4   width:100px;
5   padding:5px;
6   font-weight:bold;
7   text-align: center;
8   border:1px solid darkgray;
9 }
```

```
1 button:hover {
2   color:white;
3    background:darkgray;
4 }
5
6 button:active {
7   border:1px dashed;
8   border-color: black;
9 }
```

Popular pseudo-classes

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

```
1 button {
2   background: white;
3   color: darkgray;
4   width:100px;
5   padding:5px;
6   font-weight:bold;
7   text-align: center;
8   border:1px solid darkgray;
9 }
```

```
1 button:hover {
2   color:white;
3    background:darkgray;
4 }
5
6 button:active {
7   border:1px dashed;
8   border-color: black;
9 }
```

Popular pseudo-classes

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

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 }
```

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 }
```

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

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">Saturday's todos</p>
6   <p>Sunday’s todos</p>
7 </main>
```

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

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">Saturday's todos</p>
6   <p>Sunday’s todos</p>
7 </main>
```

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

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">Saturday's todos</p>
6   <p>Sunday’s todos</p>
7 </main>
```

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

`el1 el2`: Selects all `<el2>` elements inside `<el1>`

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">Saturday's todos</p>
6   <p>Sunday’s todos</p>
7 </main>
```

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

`el1 el2`: Selects all `<el2>` elements inside `<el1>`

whitespace in selectors implies the universal selector: `*`

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">Saturday's todos</p>
6   <p>Sunday’s todos</p>
7 </main>
```

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

`el1 el2`: Selects all `<el2>` elements inside `<el1>`

whitespace in selectors implies the universal selector: `*`

Which lines appear in orange?

```
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">Saturday's todos</p>
6   <p>Sunday’s todos</p>
7 </main>
```

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

flickr@mathiasappel

Which lines appear in orange?

```
<main>
  <h2>Todos</h2>
  <p id=“firsttodo">Today's todos</p>
  <p class=“todo">Tomorrow's todos</p>
  <p class=“todo">Saturday's todos</p>
  <p>Sunday’s todos</p>
</main>
```

```
main *:not(.todo) {
    color:orange;
}
```

Todos

Today's todos

Tomorrow's todos

Saturday's todos

Sunday’s todos

flickr@mathiasappel

Popular pseudo-classes

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

can be used to style elements with range limitations

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

```
input[type=text] {
  border: 0px;
  width: 150px;
}
input[type=number] {
  width: 100px;
}
```

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 }
```

Popular pseudo-classes

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

can be used to style elements with range limitations

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

```
input[type=text] {
  border: 0px;
  width: 150px;
}
input[type=number] {
  width: 100px;
}
```

attribute selector

Popular pseudo-classes

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

can be used to style elements with range limitations

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

```
input[type=text] {
  border: 0px;
  width: 150px;
}
input[type=number] {
  width: 100px;
}
```

attribute selector

```
input:valid + label::after {
 content:" \2714";
 color: rgba(0,100,0,0.7);
}

input:invalid + label::after {
 content:" (invalid)";
 color: rgba(255,0,0,0.7);
}
```

Popular pseudo-classes

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

can be used to style elements with range limitations

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

```
input[type=text] {
  border: 0px;
  width: 150px;
}
input[type=number] {
  width: 100px;
}
```

attribute selector

```
input:valid + label::after {
 content:" \2714";
 color: rgba(0,100,0,0.7);
}

input:invalid + label::after {
 content:" (invalid)";
 color: rgba(255,0,0,0.7);
}
```

adjacent selector

Popular pseudo-classes

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

can be used to style elements with range limitations

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

```
input[type=text] {
  border: 0px;
  width: 150px;
}
input[type=number] {
  width: 100px;
}
```

attribute selector

adjacent selector

```
input:valid + label::after {
 content:" \2714";
 color: rgba(0,100,0,0.7);
}

input:invalid + label::after {
 content:" (invalid)";
 color: rgba(255,0,0,0.7);
}
```

rgb & alpha

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

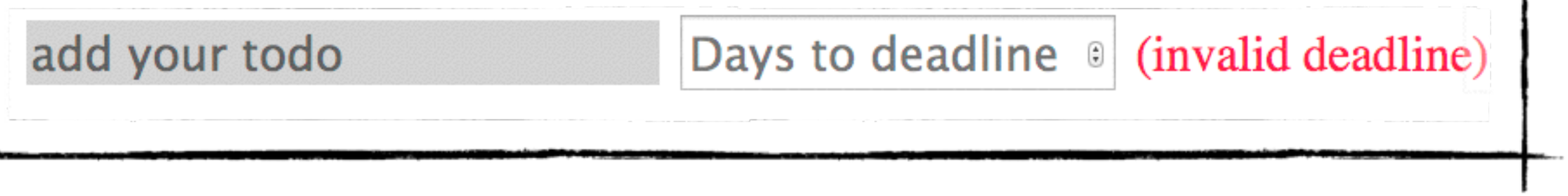
`:in-range` `:out-of-range`

can be used to style elements with range limitations

Popular pseudo-classes

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

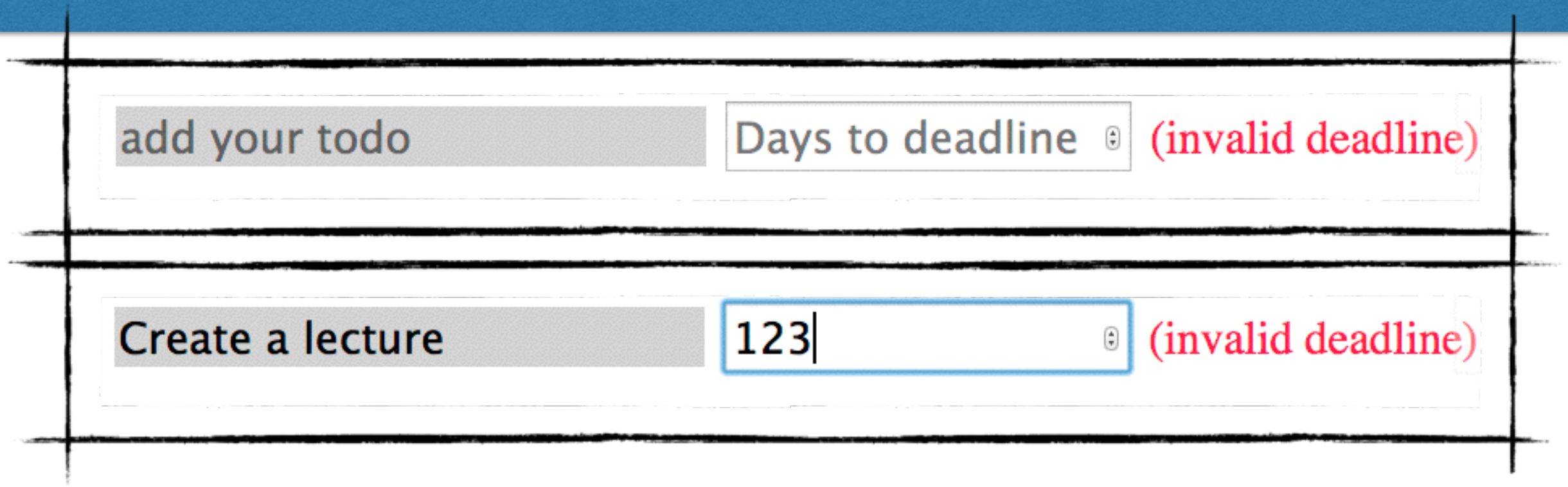
can be used to style elements with range limitations



Popular pseudo-classes

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

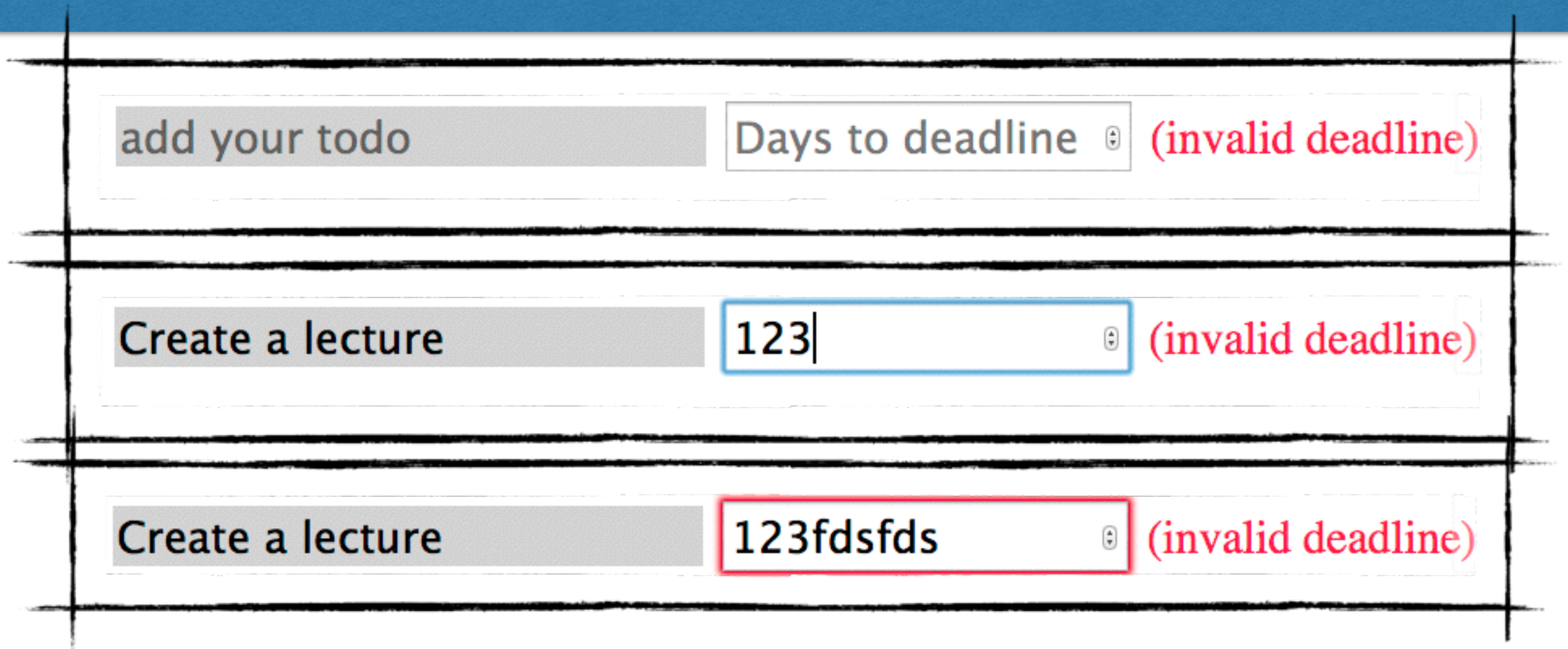
can be used to style elements with range limitations



Popular pseudo-classes

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

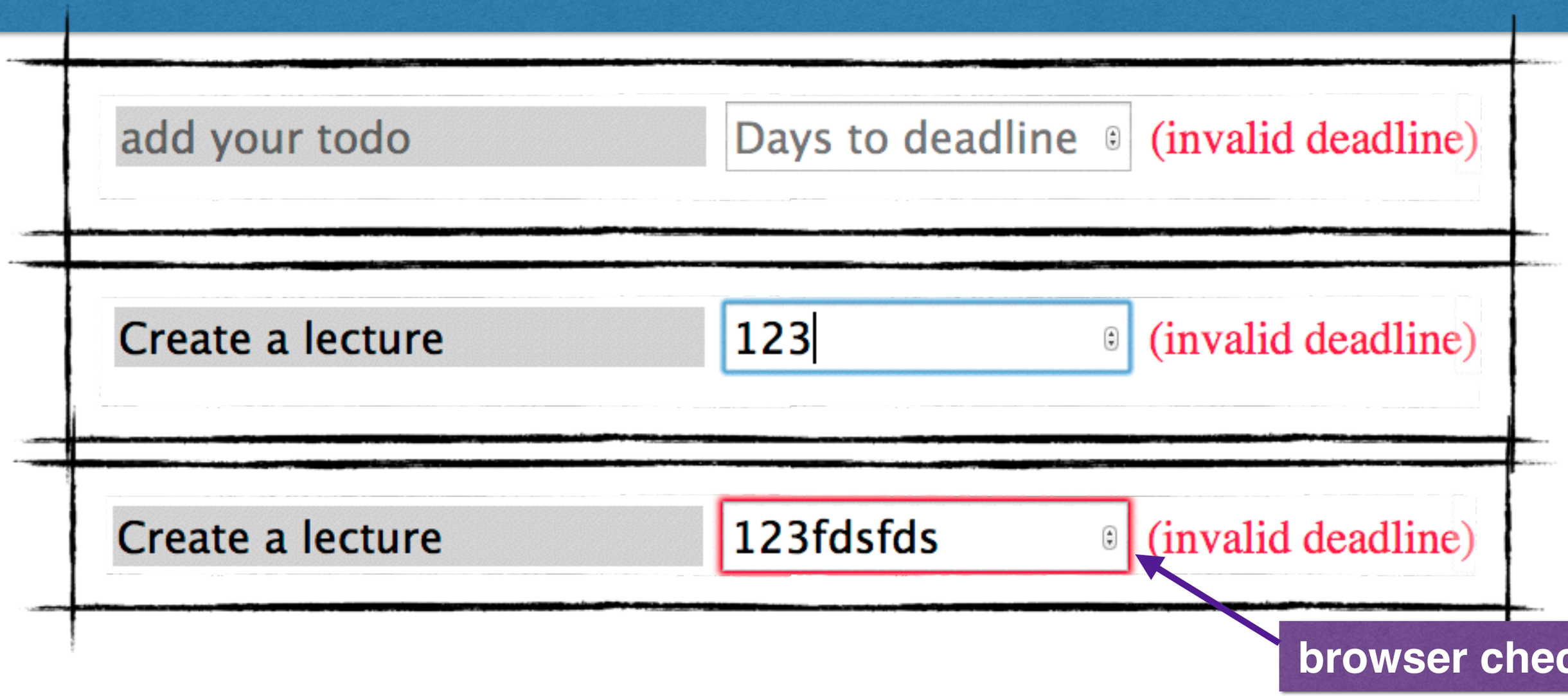
can be used to style elements with range limitations



Popular pseudo-classes

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

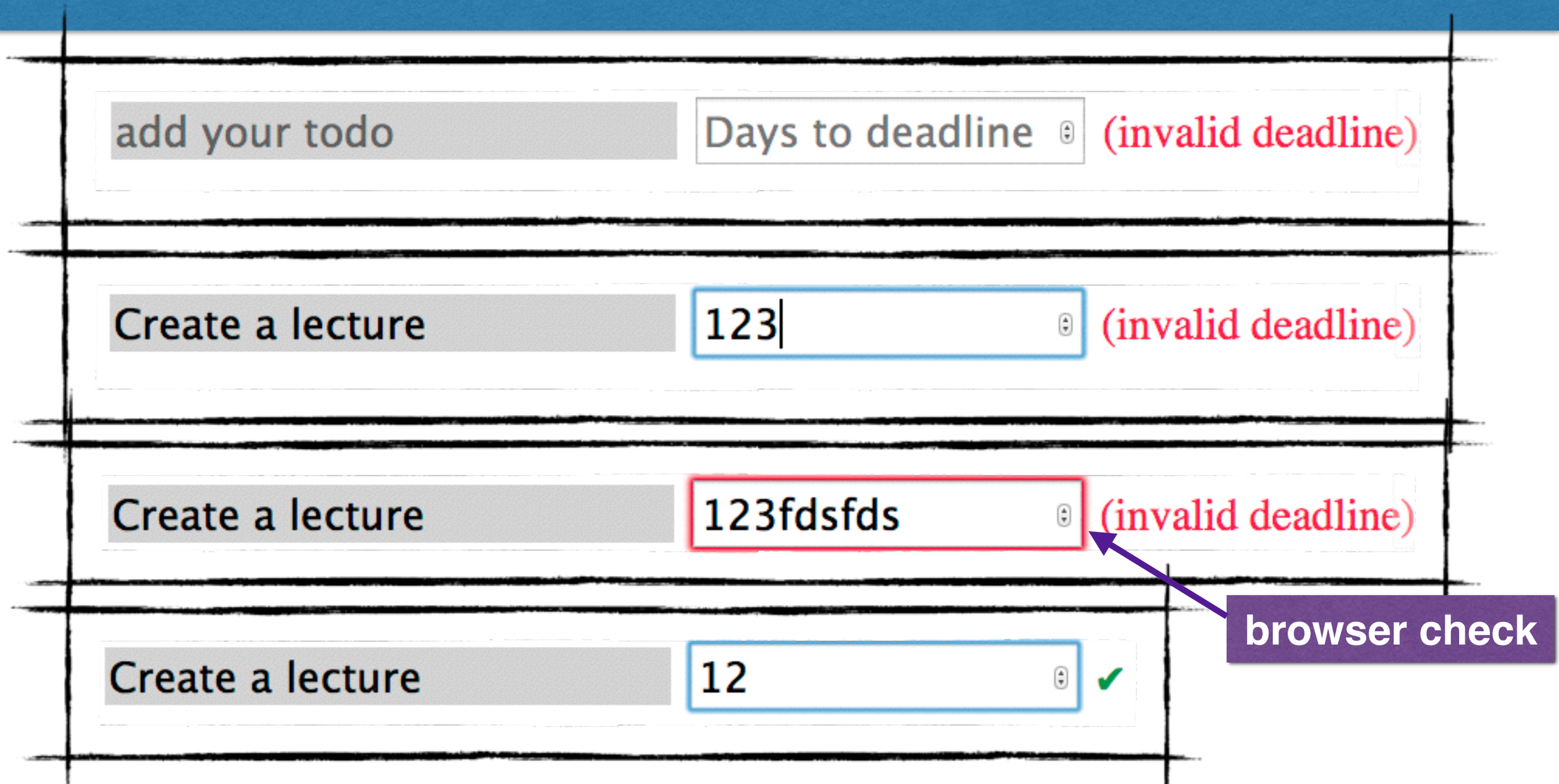
can be used to style elements with range limitations



Popular pseudo-classes

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

can be used to style elements with range limitations



Pseudo-elements

Pseudo-elements

Pseudo-element: creates an abstractions about the document tree beyond those specified by the document language.

Pseudo-elements

Pseudo-element: creates an abstractions about the document tree beyond those specified by the document language.

:: notation, but … one-colon notation also acceptable for older pseudo-elements

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 }
```

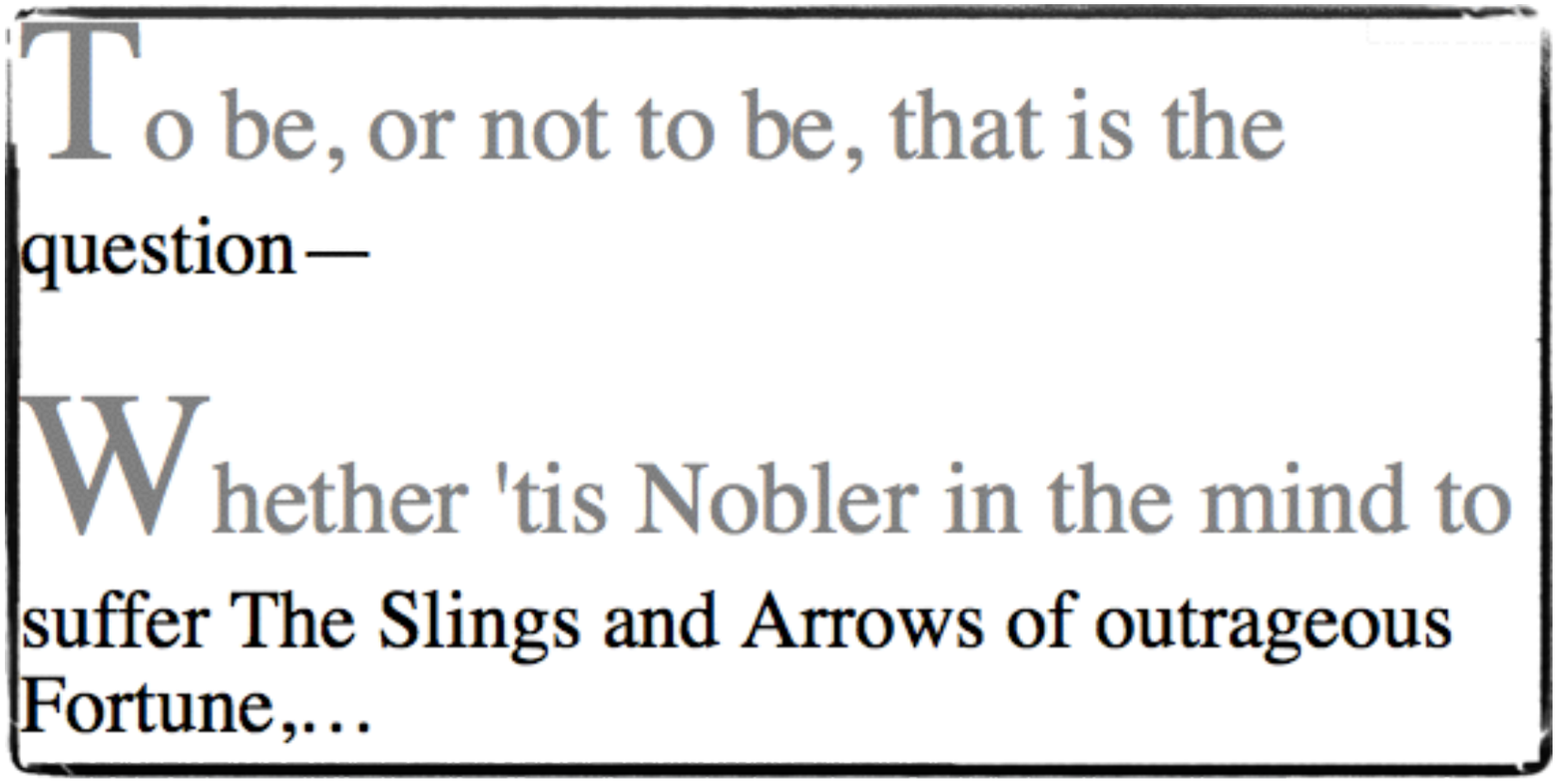
```
1 <p>
2   To be, or not to be, that
3   is the question—
4 </p>
5 <p>
6   Whether 'tis Nobler in the
7   mind to suffer
8   The Slings and Arrows of
9   outrageous Fortune,…
10 </p>
```

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

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

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

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

```
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

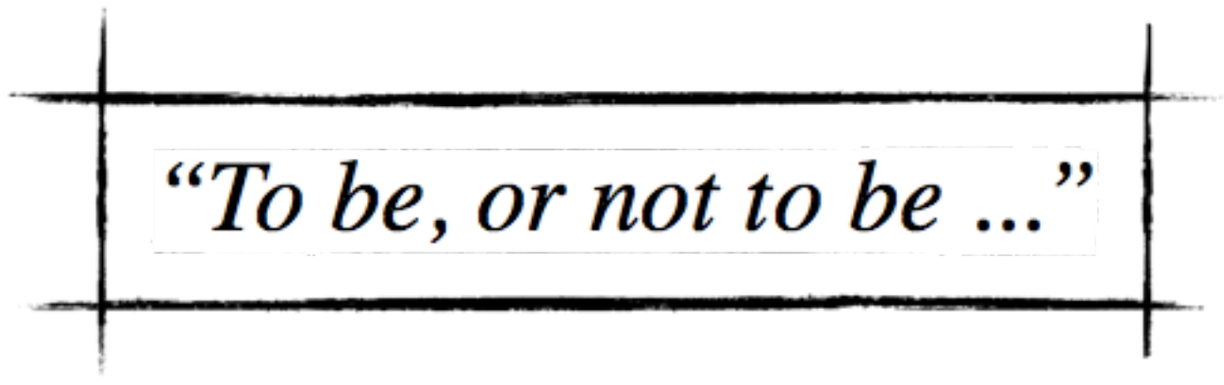
`::after` used to add (cosmetic) content after an element

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

```
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



Will this work?

```
<cite>
</cite>
```

```
cite::before {
    content: "\201CTo be, or ";
}
cite::after {
    content: "not to be ... \201D";
}
```

flickr@nathaninsandiego

Data in CSS

CSS & data (one way)

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

CSS & data (one way)

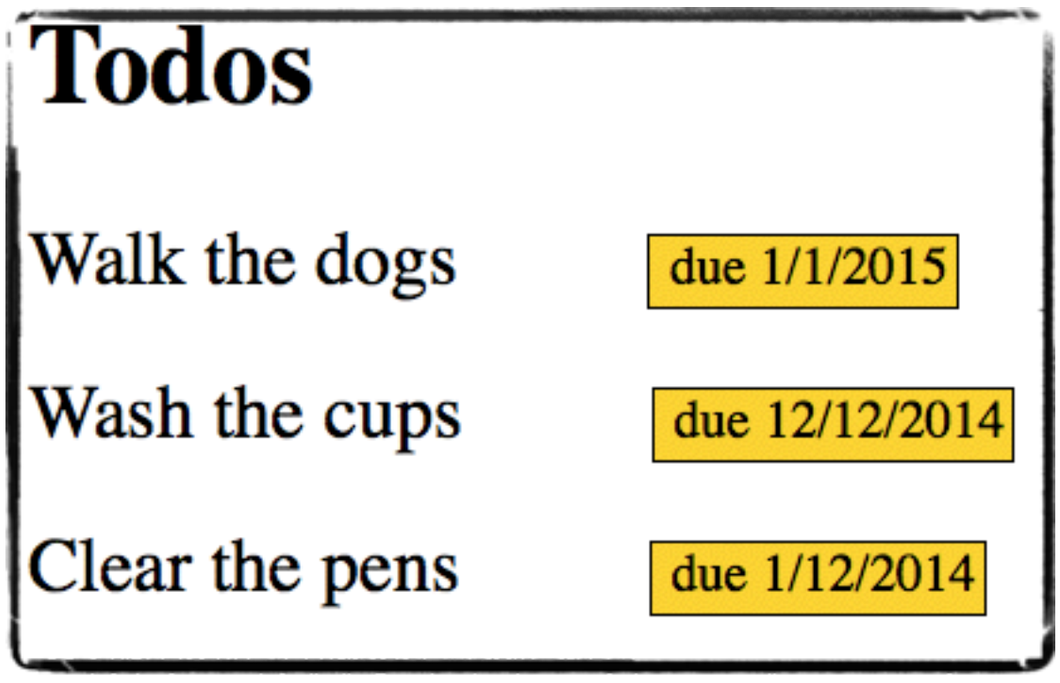
CSS does not only describe the style, it can carry data too.

```
<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 & data (one way)

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

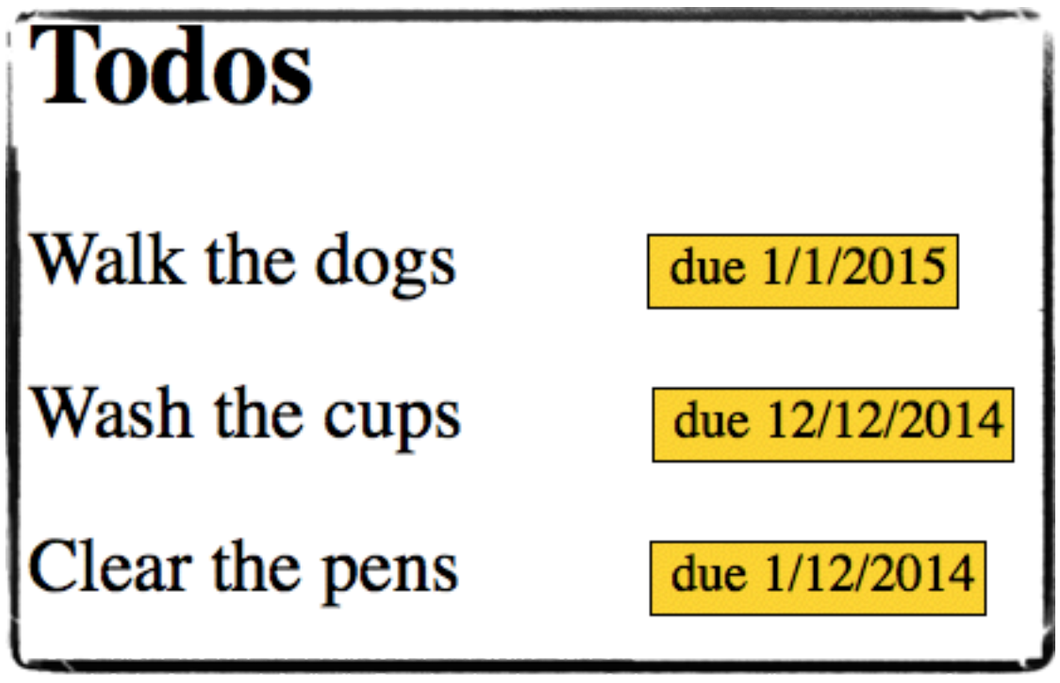
```
1 <main>
2   <h2>Todos</h2>
3     <p id="t1">Walk the dogs</p>
4     <p id="t2">Wash the cups</p>
5     <p id="t3">Clear the pens</p>
6 </main>
```



CSS & data (one way)

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

```
<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>
```




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

p#t1::after {
    content: " due 1/1/2015";
}

p#t2::after {
    content: " due 12/12/2014";
}

p#t3::after {
    content: " due 1/12/2014";
}
```

CSS & data (one way)

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

```
<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>
```

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

p#t3::after {
    content: " due 1/12/2014";
}
```

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)

Walk the dogs
Wash the cups
Clear the pens due 1/12/2014

CSS & data-* (the preferred way)

CSS can make use of **data stored in HTML elements**.

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

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

CSS & data-* (the preferred way)

CSS can make use of **data stored in HTML elements**.

Recall: HTML elements can have `data-*` attributes.

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

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

CSS & data-* (the preferred way)

CSS can make 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

CSS & data-* (the preferred way)

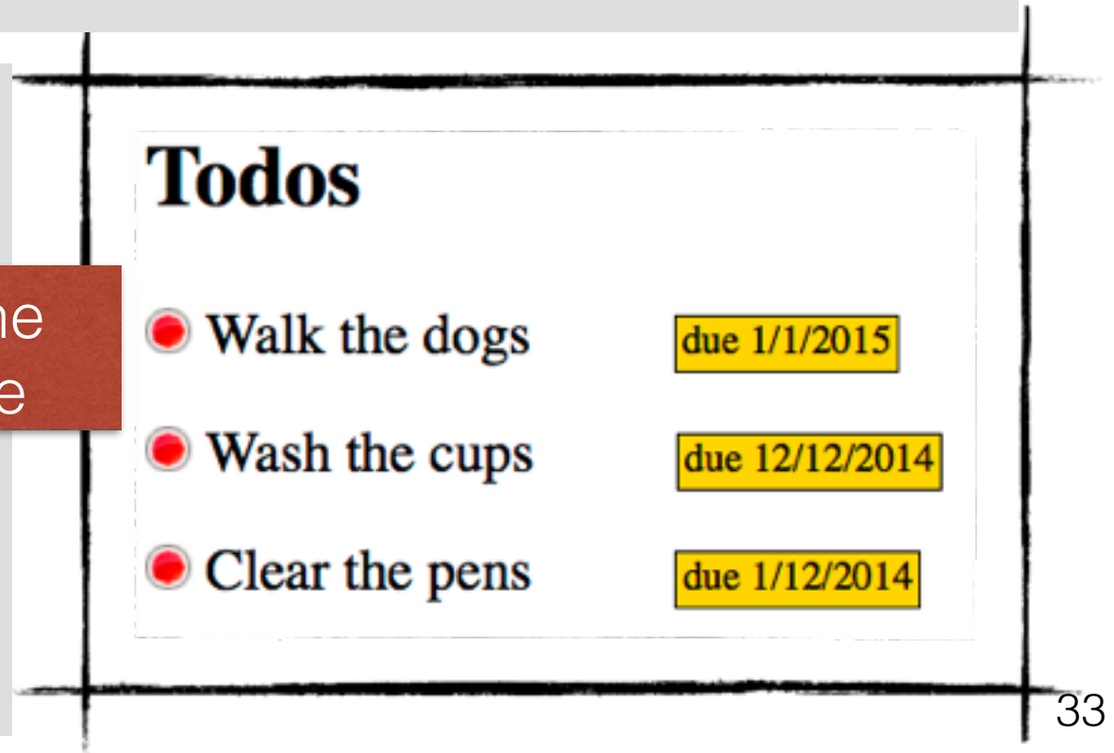
CSS can make use of **data stored in HTML elements**.

Recall: HTML elements can have `data-*` attributes.

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

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

`attr()` retrieves the value of an attribute



CSS & data-* (the preferred way)

CSS can make use of **data stored in HTML elements**.

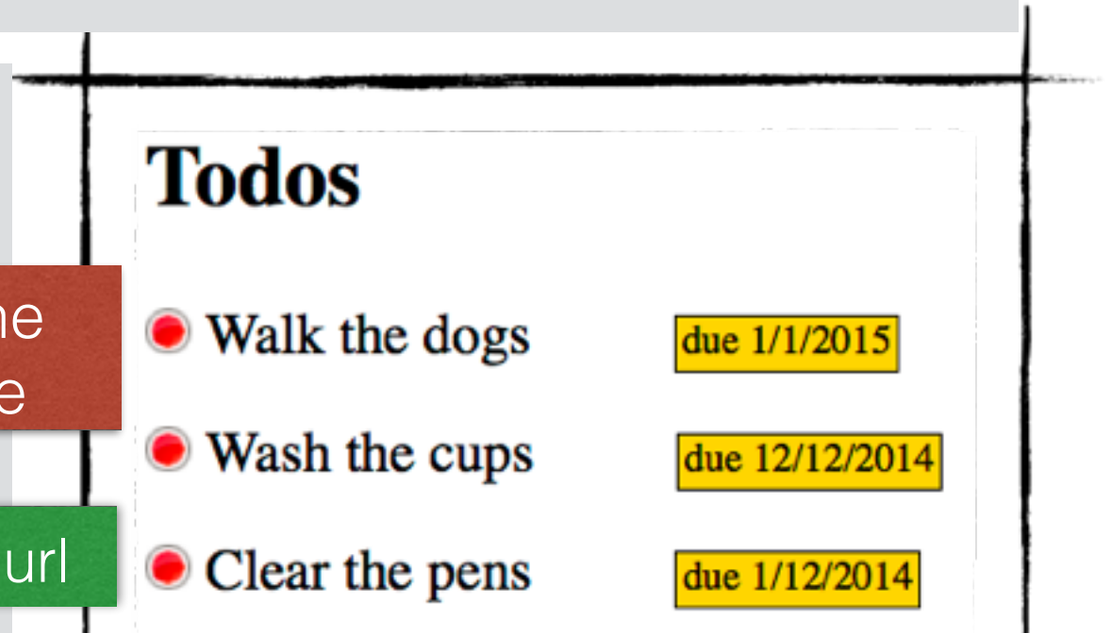
Recall: HTML elements can have `data-*` attributes.

```
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::bef
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)

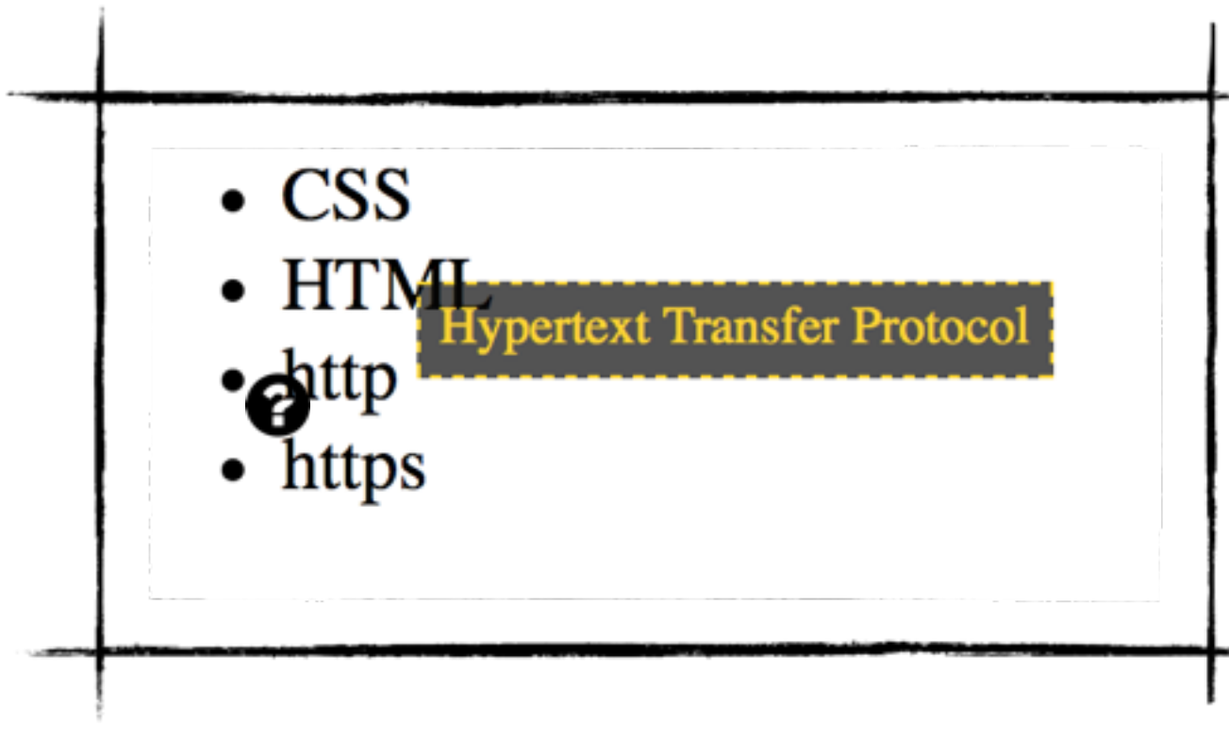
Another example: a simple tooltip

```
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>
```

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>
```

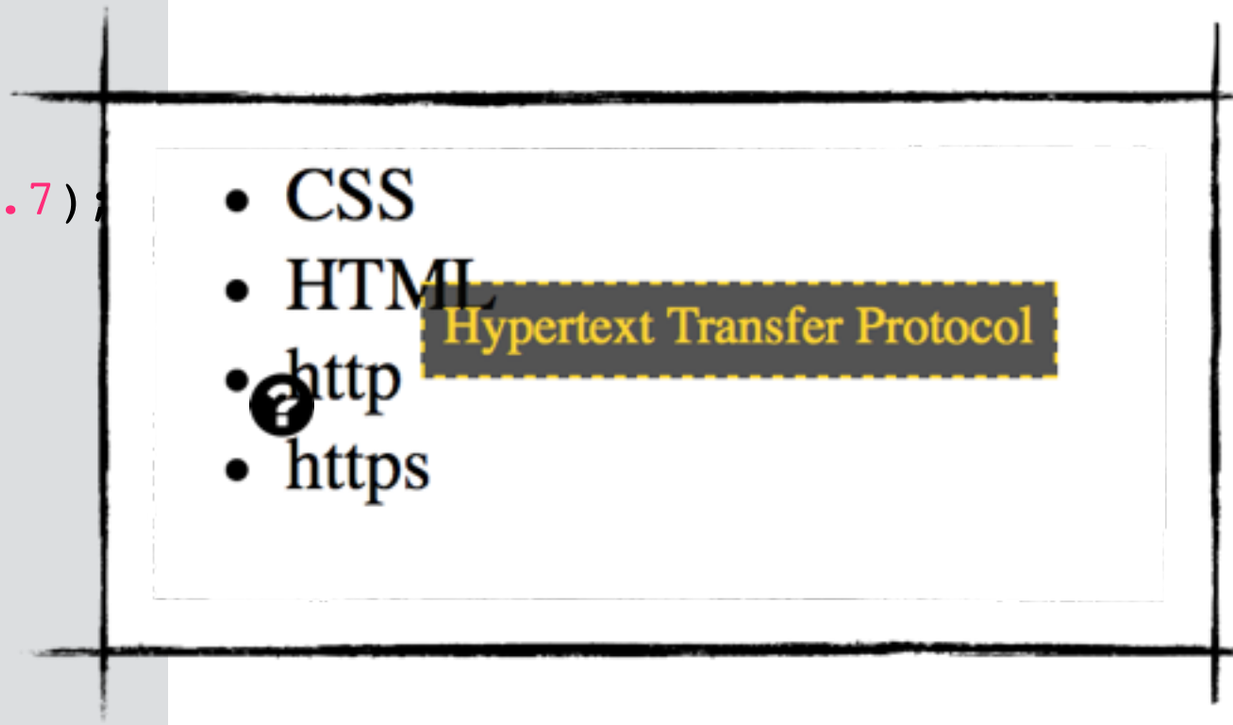


CSS & data-* (the preferred way)

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 }
```



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

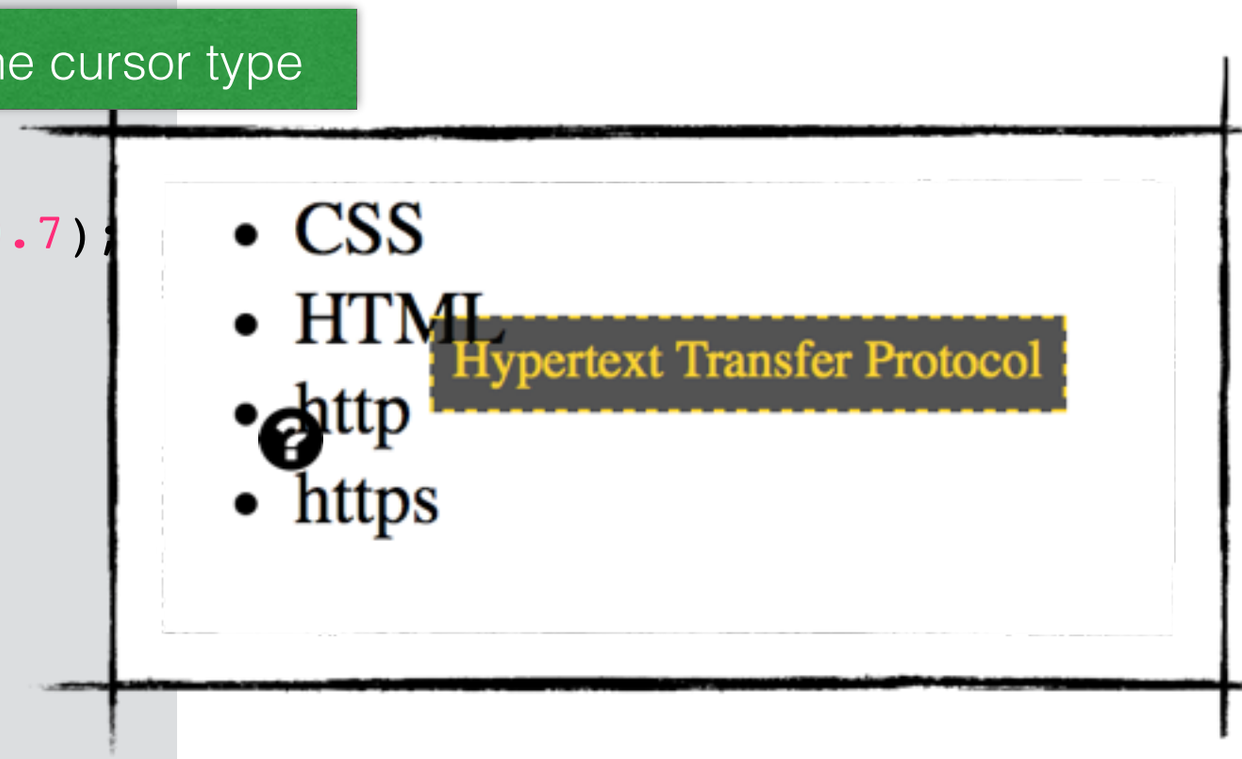
CSS & data-* (the preferred way)

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 counters

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

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>
```

CSS counters

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

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

```
body {
    /* initialize counter to 0 */
    counter-reset: countTodo;
}
p::before {
    /* increment at each <p> */
    counter-increment: countTodo;
    /* counter written out */
    content:" Todo " counter(countTodo)": ";
}
```

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,".") ": ";
}
```

Nested CSS counters

Child elements receive their own counter instance.

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

```html
<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>
```

```css
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

How are the list elements enumerated?

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

<p>Tomorrow's todos</p>
<ul>
  <li>Walk the dogs</li>
  <li>Wash the dishes</li>
</ul>
```

```
body {
  counter-reset: cli;
}

ul {
   list-style: none;
}

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

flickr@darrenputtock

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**

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).

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).

```
main:-webkit-full-screen {
} /* Chrome */

main:-moz-full-screen {
} /* Firefox */

main:-ms-fullscreen {
} /* Internet Explorer */

main:fullscreen {
} /* W3C proposal */
```

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).

```
main:-webkit-full-screen {
} /* Chrome */

main:-moz-full-screen {
} /* Firefox */

main:-ms-fullscreen {
} /* Internet Explorer */

main:fullscreen {
} /* W3C proposal */
```

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

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).

Recent move towards disabling experimental features in browsers by default; explicit reset by user required.

But … vendor prefixes will not go away soon (that would break a lot of pages on the Web)

```
8 } /* Internet Explorer */
9
10 main:fullscreen {
11 } /* W3C proposal */
```

browser release might break the implemented CSS

-webkit? Google Chrome is not based on Webkit anymore …

Will we see a `-chrome-` vendor prefix now?

We've seen how the proliferation of vendor prefixes has caused pain for developers and we don't want to exacerbate this. As of today, Chrome is adopting a policy on vendor prefixes, one that is similar to Mozilla's recently announced policy.

In short: we won't use vendor prefixes for new features. Instead, we'll expose a single setting (in `about:flags`) to enable experimental DOM/CSS features for you to see what's coming, play around, and provide feedback, much as we do today with the "Experimental WebKit Features"/"Enable experimental Web Platform features" flag. Only when we're ready to see these features ship to stable will they be enabled by default in the dev/canary channels.

For legacy vendor-prefixed features, we will continue to use the `-webkit-` prefix because renaming all these prefixes to something else would cause developers unnecessary pain. We've started looking into real world usage of HTML5 and CSS3 features and hope to use data like this to better inform how we can responsibly deprecate prefixed properties and APIs. As for any non-standard features that we inherited (like `-webkit-box-reflect`), over time we hope to either help standardize or deprecate them on a case-by-case basis.

http://www.chromium.org/blink/developer-faq

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

Elements “flow” by default

```
<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>`

```
<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>`

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

HTML 4 terminology

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>`

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

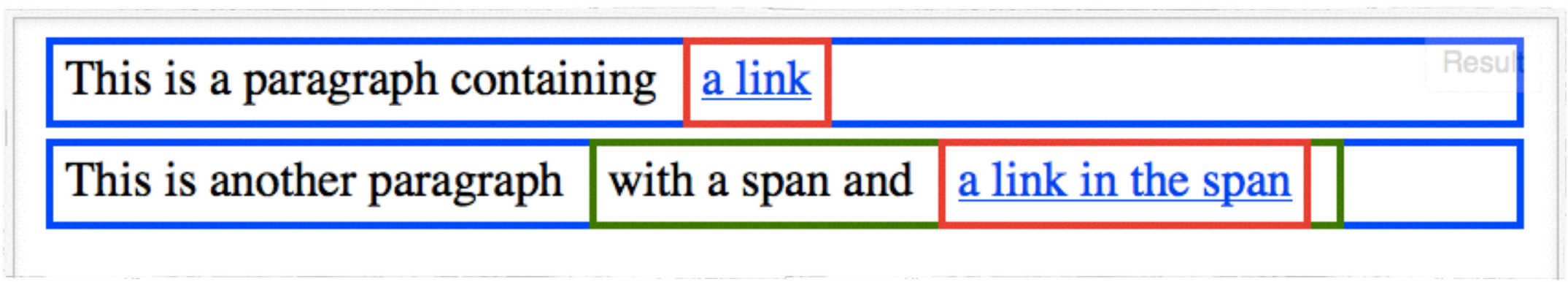
HTML 4 terminology

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>`

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: auto;}
```

Elements “flow” by default

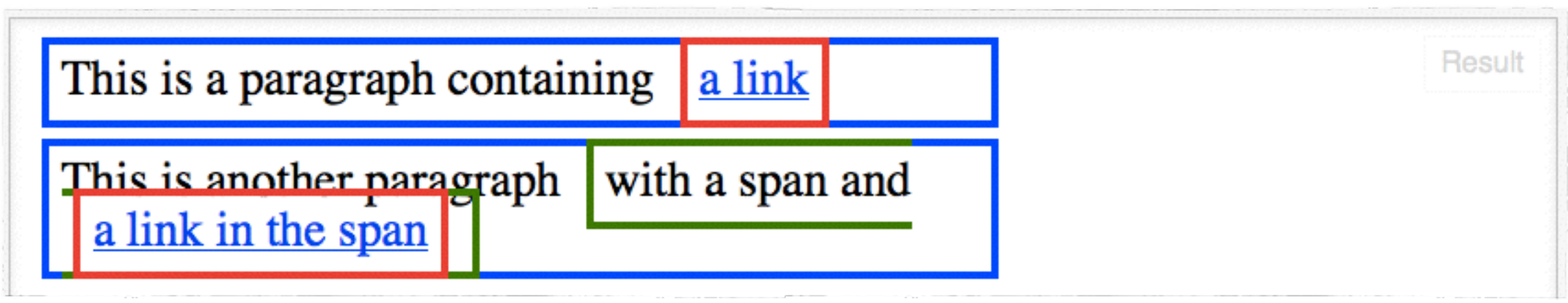
HTML 4 terminology

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>`

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>`



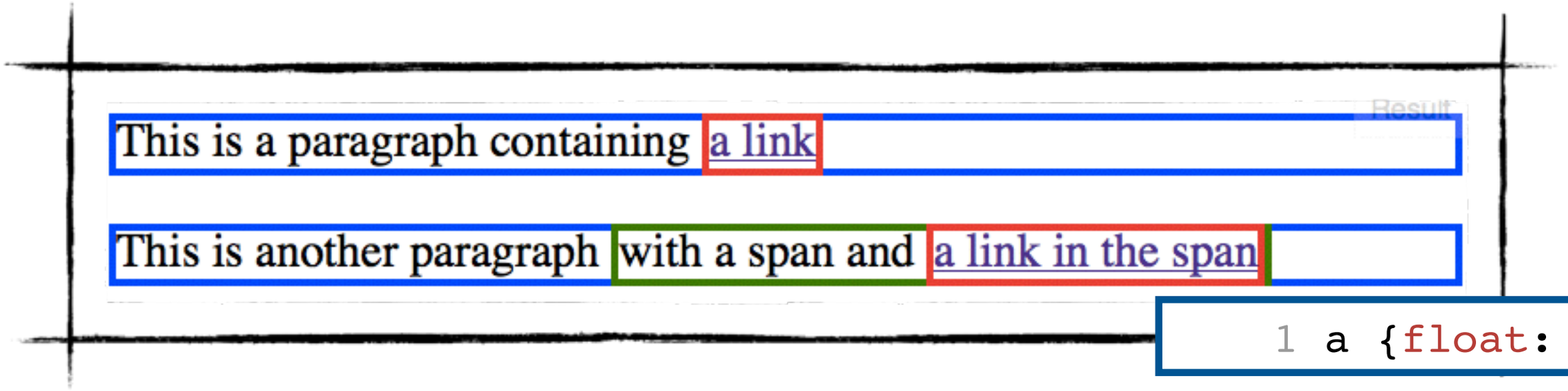

```
1 main {width: 400px;}
```

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.

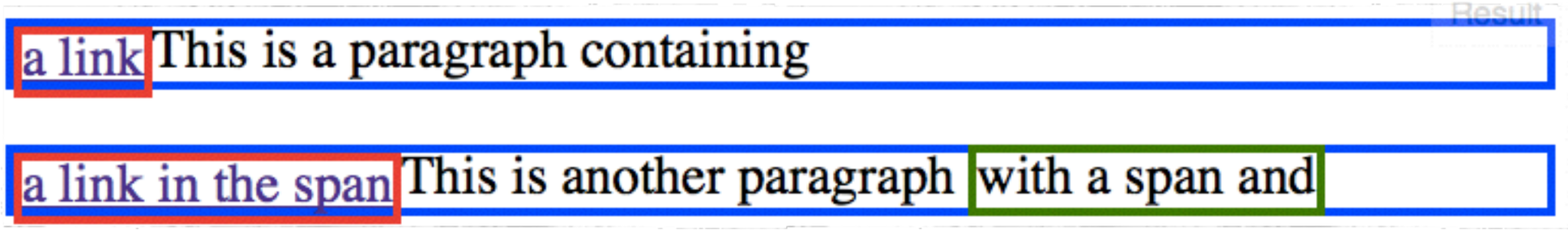
Taking elements out of the flow

`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.



Taking elements out of the flow

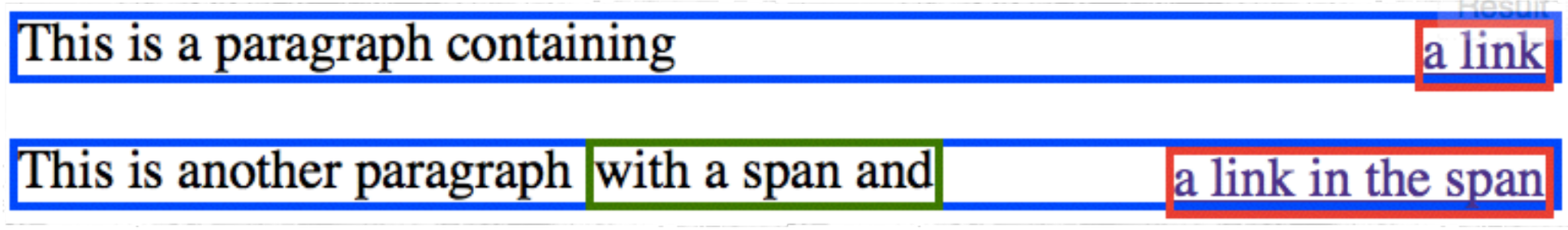
`float:left` (or **`:right`**) takes an element out of the flow; it is moved to the leftmost (or rightmost) possible position **in the containing element** —either the element edge or another float.




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

Taking elements out of the flow

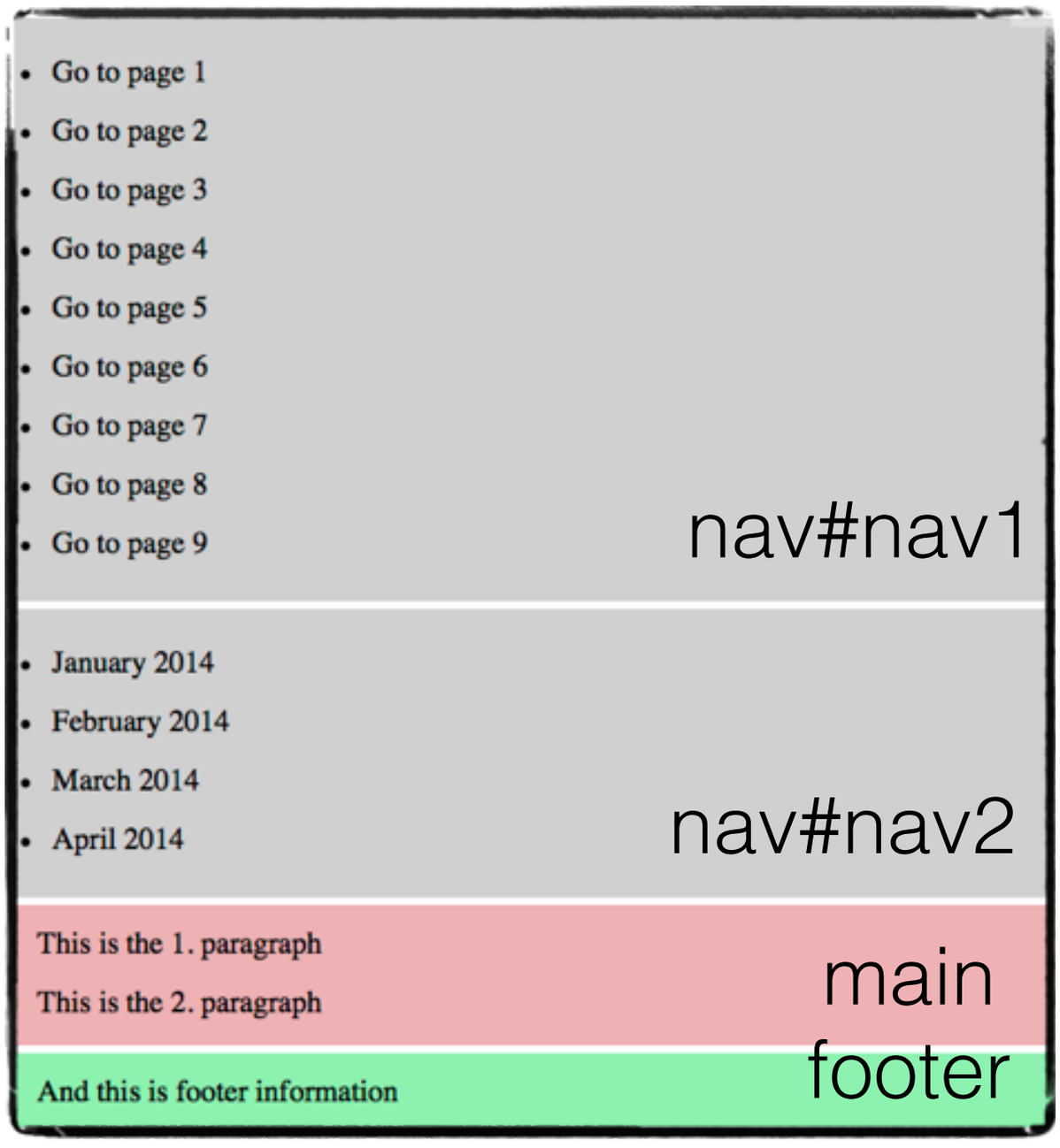
`float:left` (or **`:right`**) takes an element out of the flow; it is moved to the leftmost (or rightmost) possible position **in the containing element** —either the element edge or another float.




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

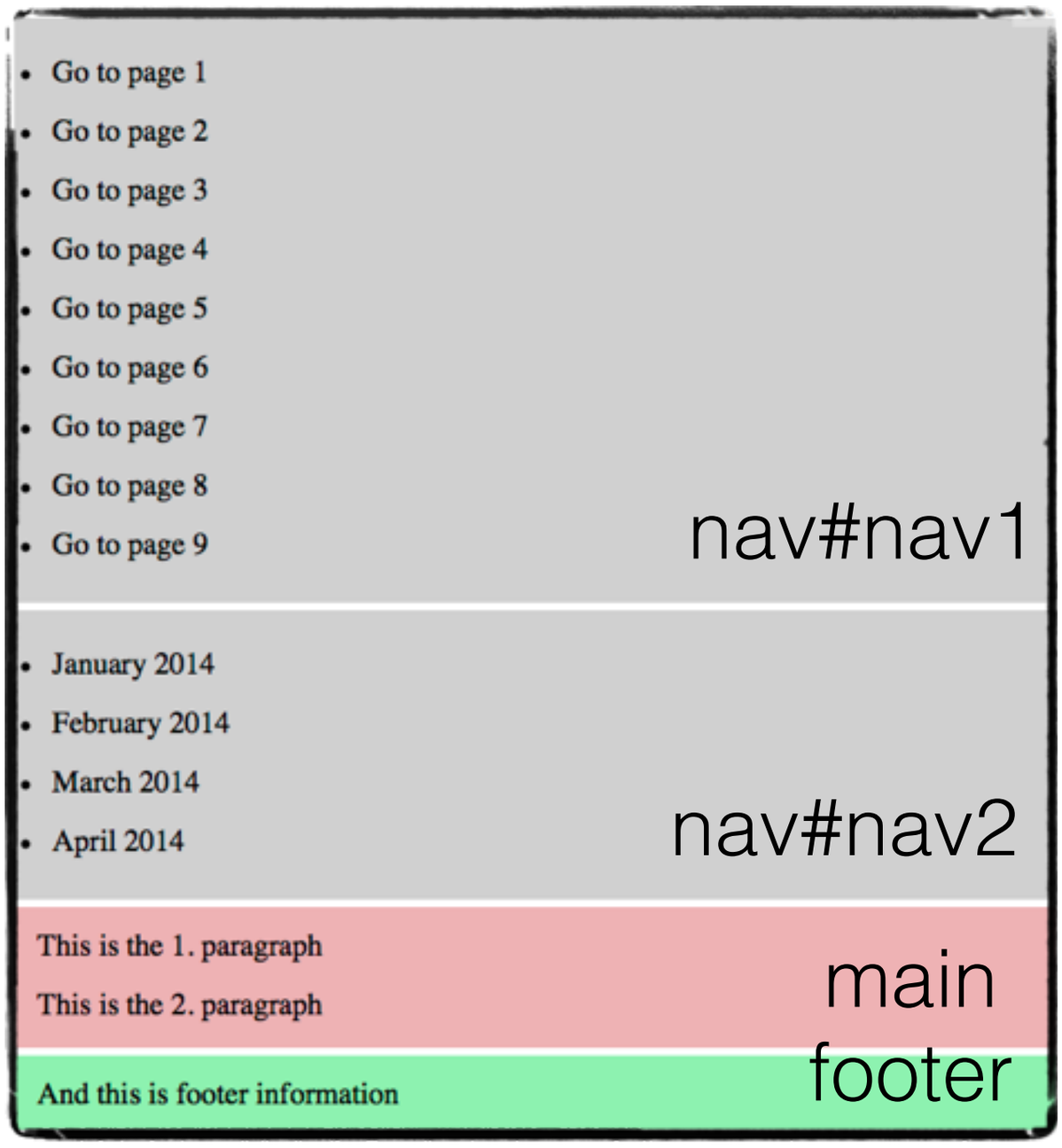
Resetting the flow with `clear`

Canonical example: **adding sidebars** to a layout



Resetting the flow with `clear`

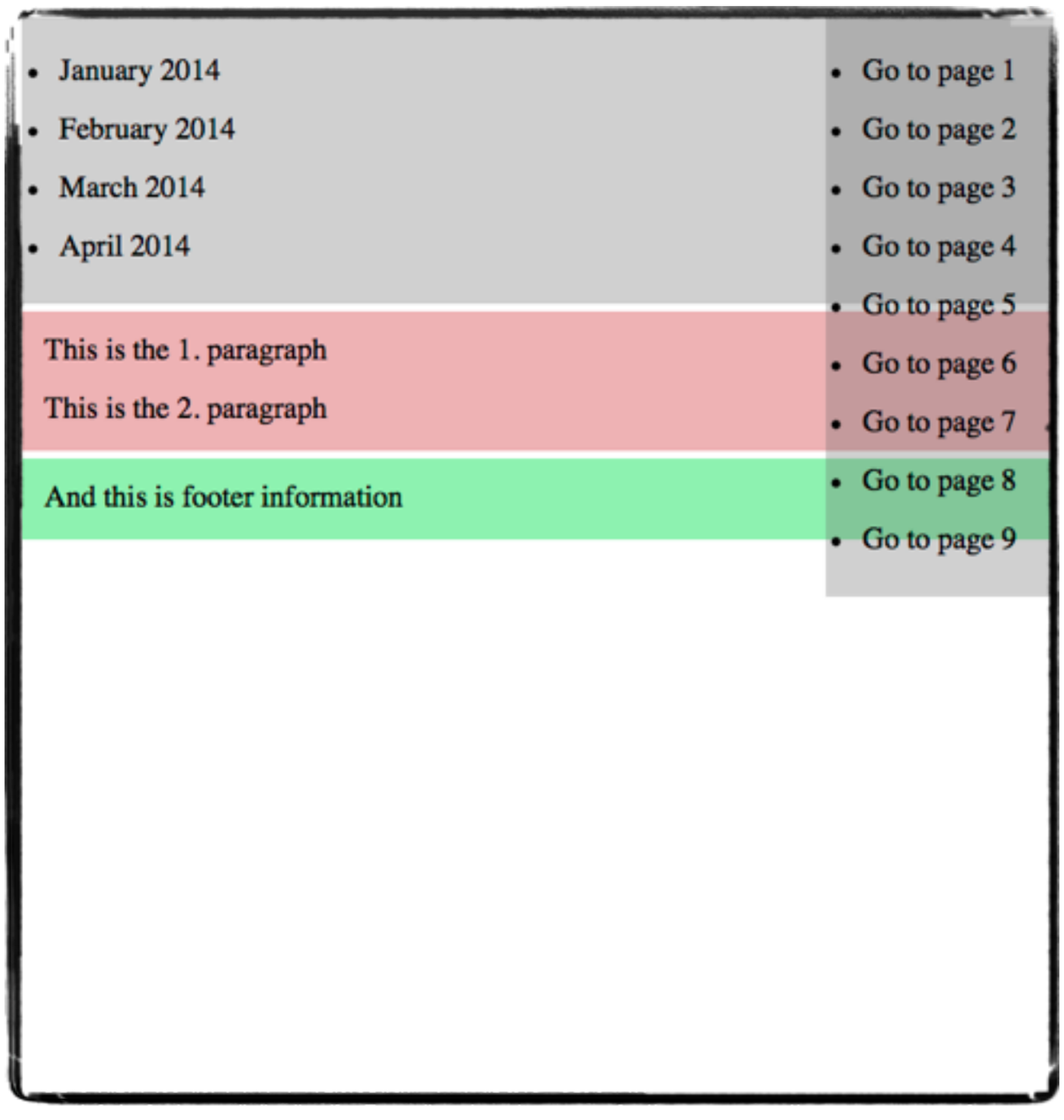
Canonical example: **adding sidebars** to a layout




```
1 #nav1 {float: right;}
```

Resetting the flow with `clear`

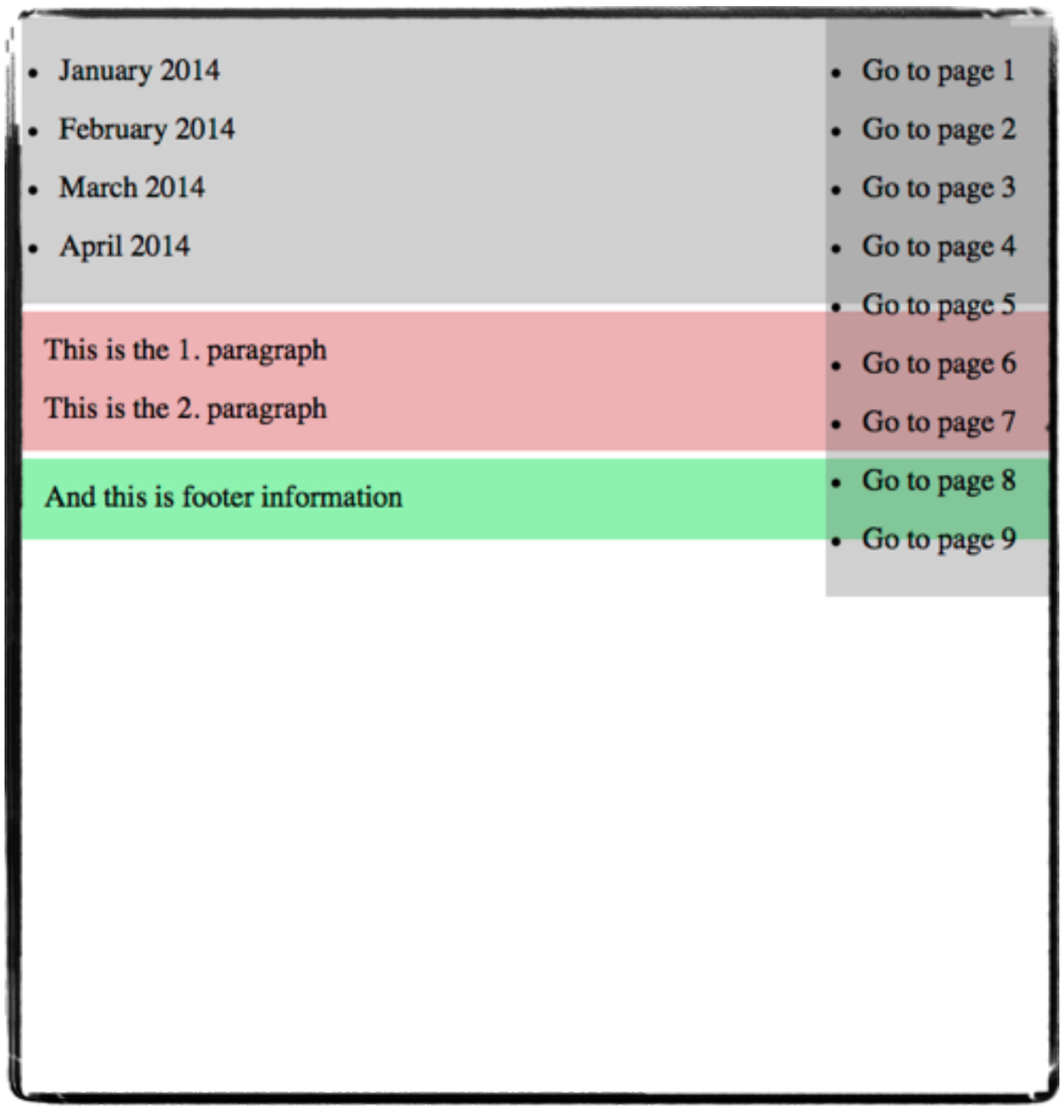
Canonical example: **adding sidebars** to a layout




```
1 #nav1 {float: right;}
```

Resetting the flow with `clear`

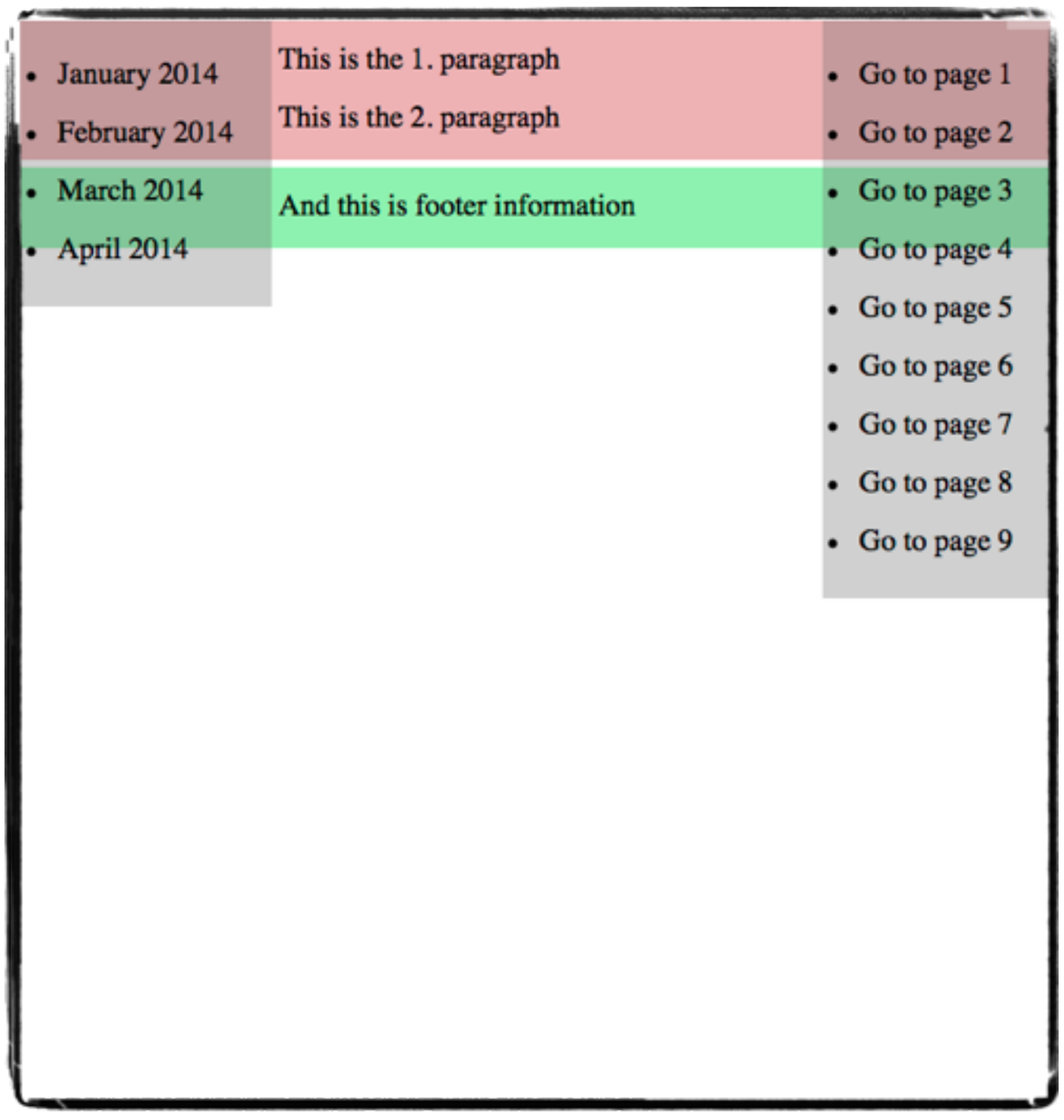
Canonical example: **adding sidebars** to a layout




```
1 #nav1 {float: right;}
2 #nav2 {float: left;}
```

Resetting the flow with `clear`

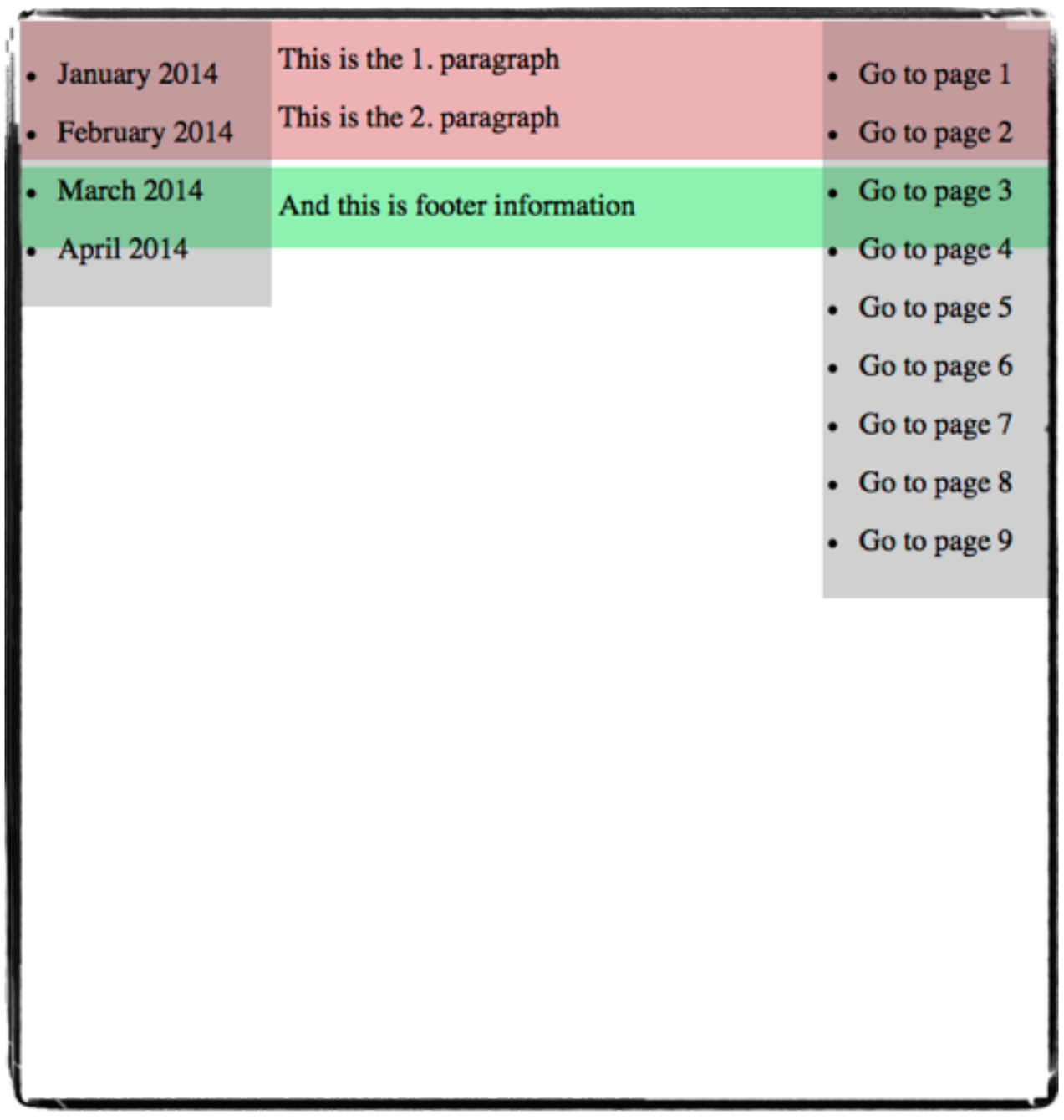
Canonical example: **adding sidebars** to a layout




```
1 #nav1 {float: right;}
2 #nav2 {float: left;}
```

Resetting the flow with `clear`

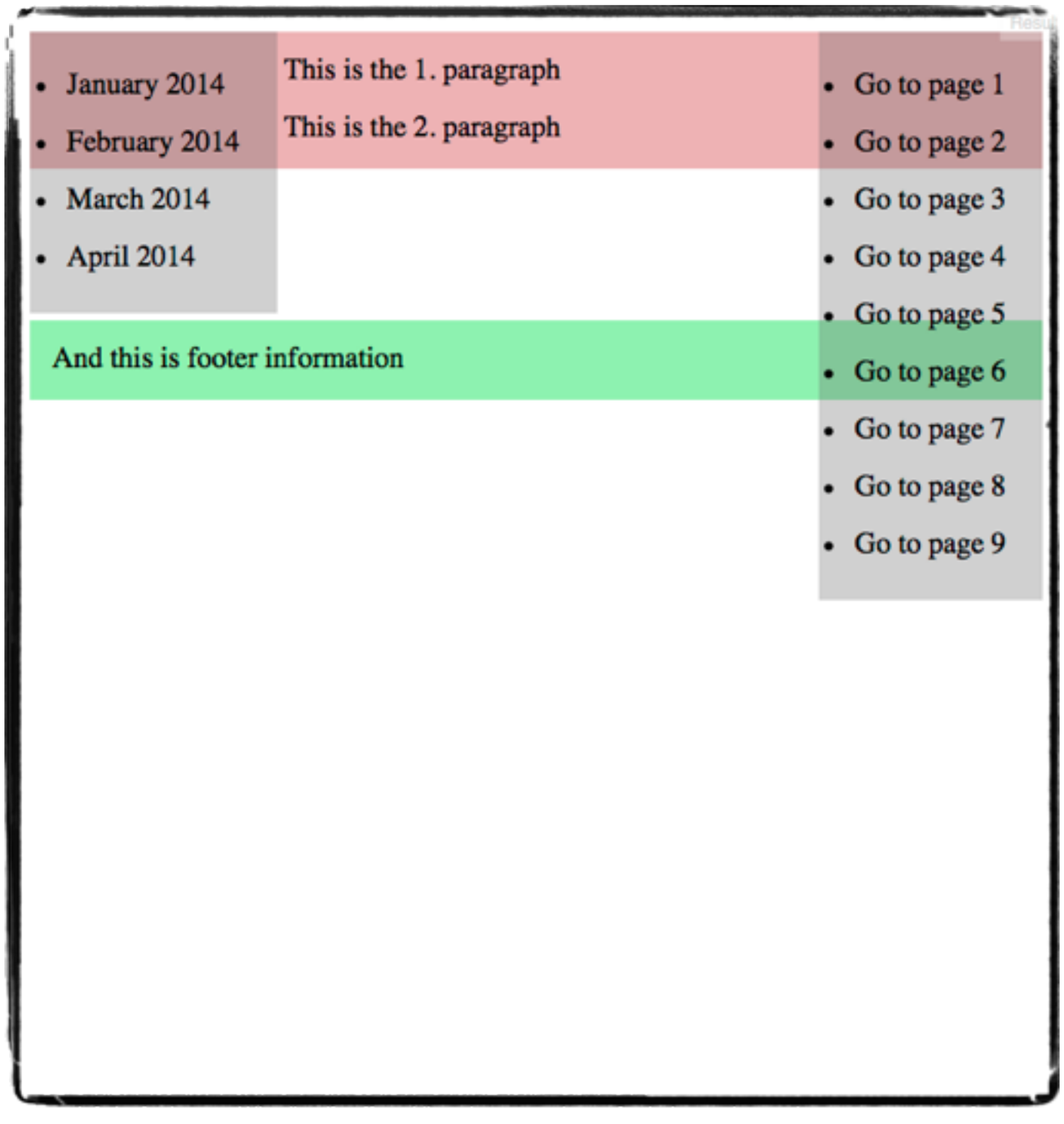
Canonical example: **adding sidebars** to a layout




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

Resetting the flow with `clear`

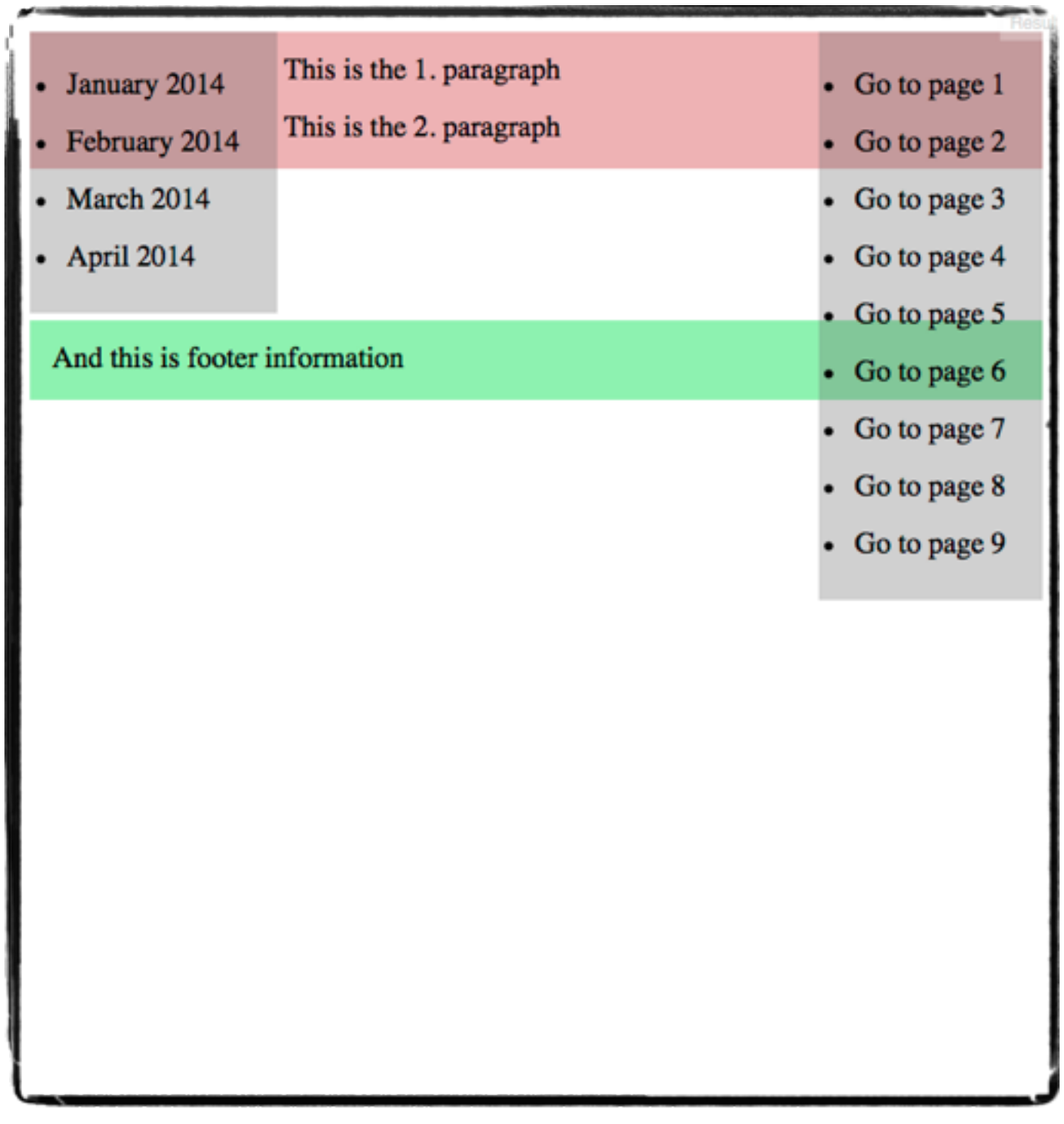
Canonical example: **adding sidebars** to a layout




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

Resetting the flow with `clear`

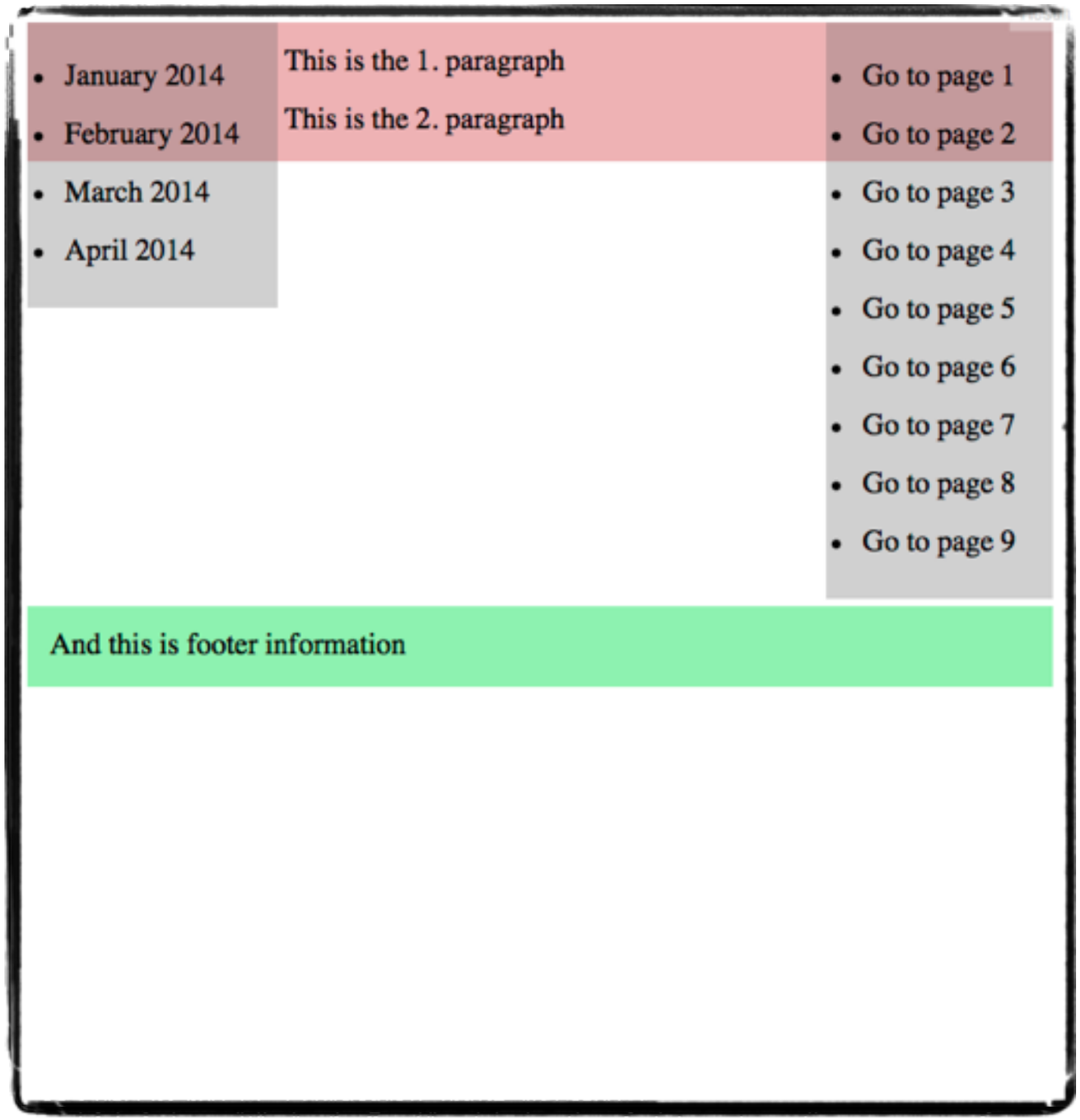
Canonical example: **adding sidebars** to a layout




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

Resetting the flow with `clear`

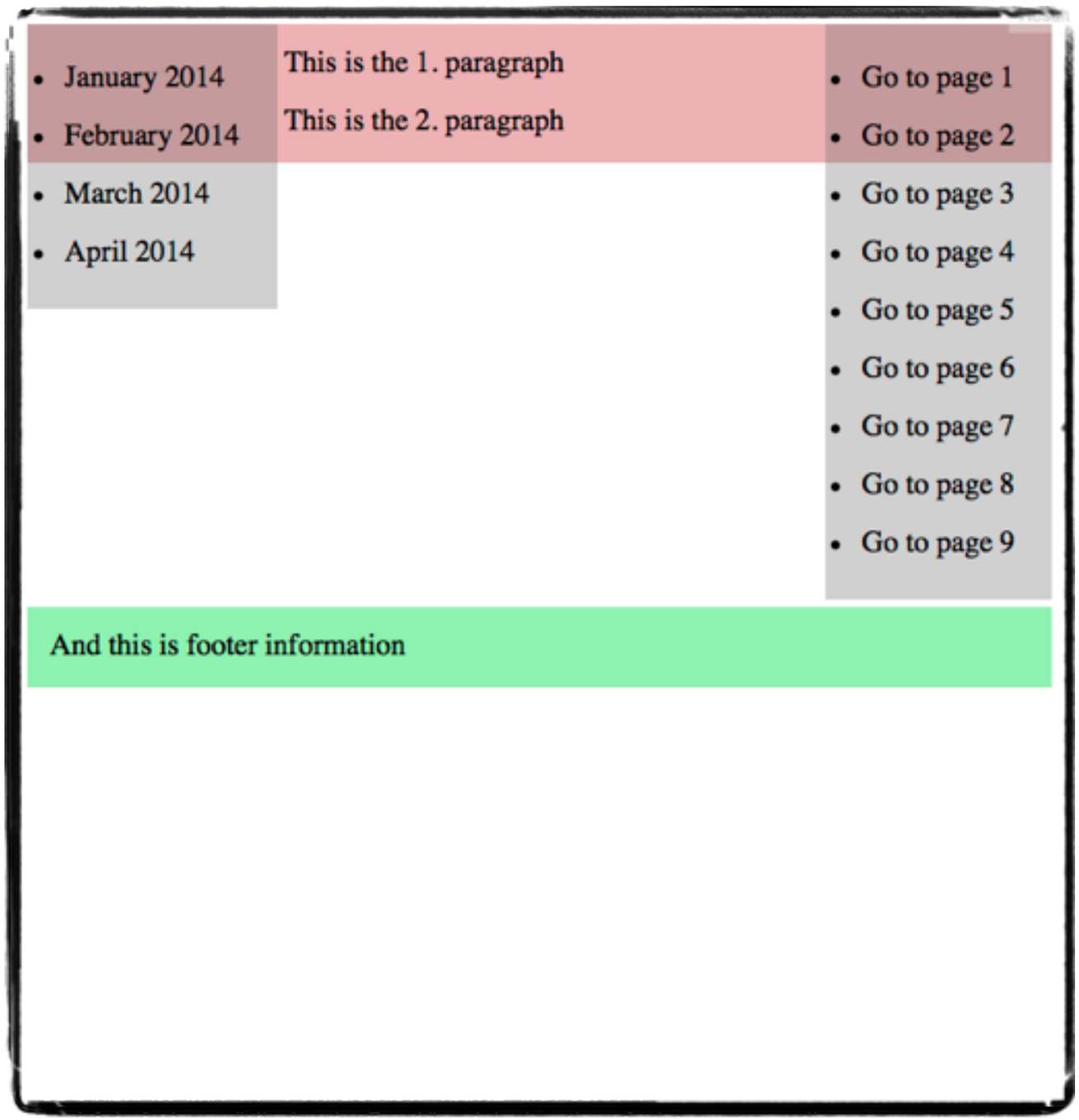
Canonical example: **adding sidebars** to a layout




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

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.

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.

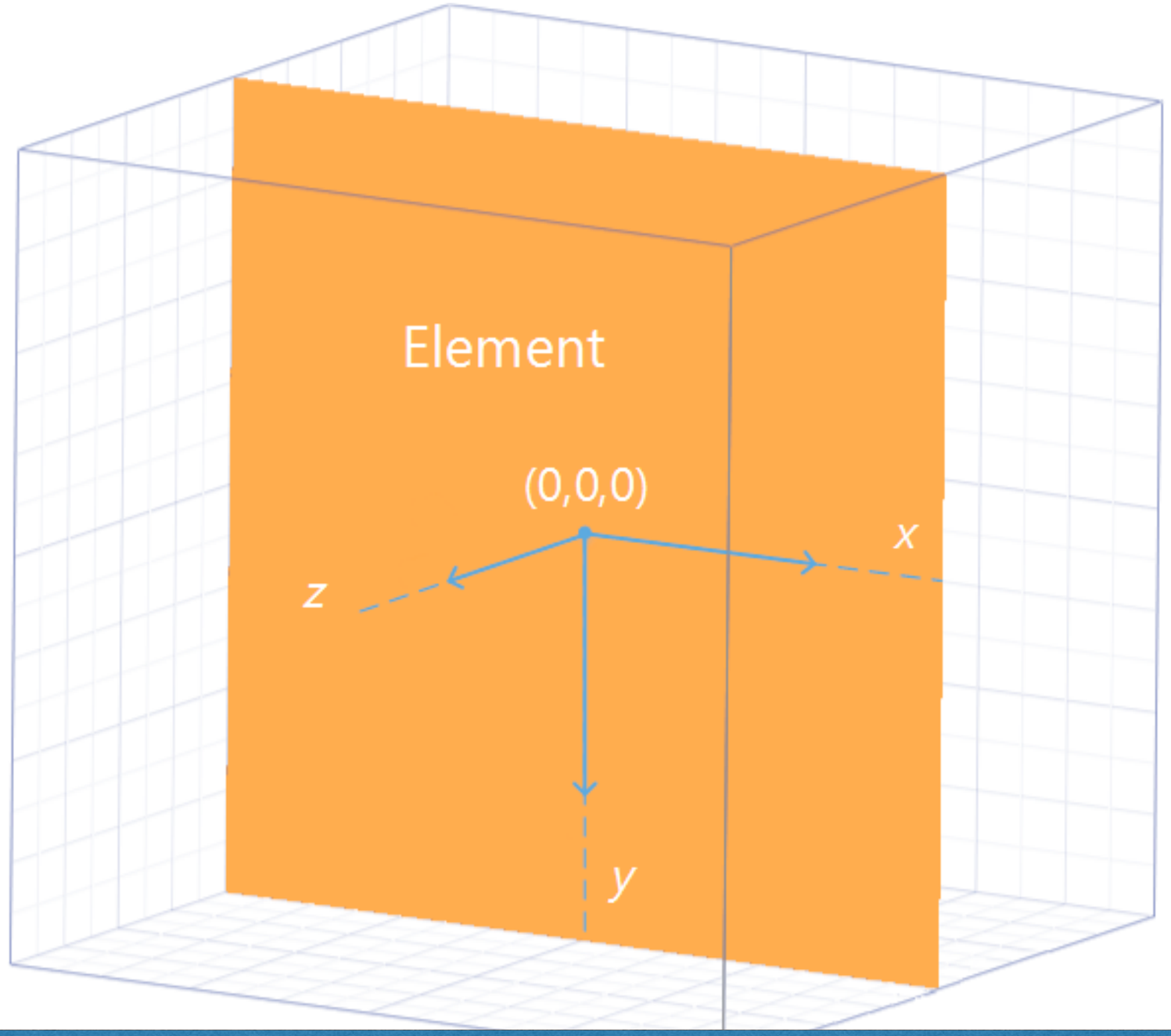
`position:static`	the default
`position:relative`	the element is adjusted on the fly, other elements are not affected
`position:absolute`	element is taken out of the normal flow (no space is reserved for it)
`position:fixed`	similar to absolute, but fixed to the **viewport**
`position:sticky`	in-between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

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`	**the area currently being viewed** the **viewport**
`position:sticky`	in-between relative and fixed

CSS coordinate system



y extends downward. **x** extends to the right.

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

position:relative

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

movement is **relative** to its original position

id="egg1"

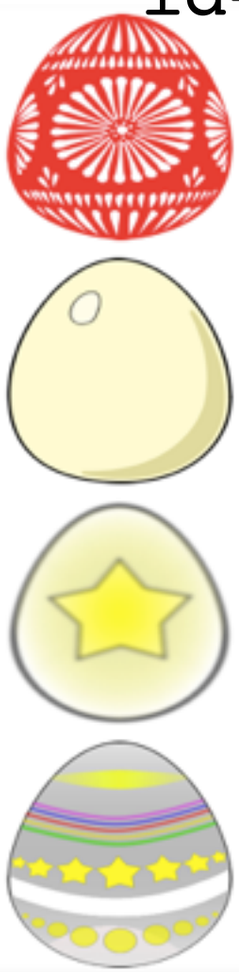
id="egg4"

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"

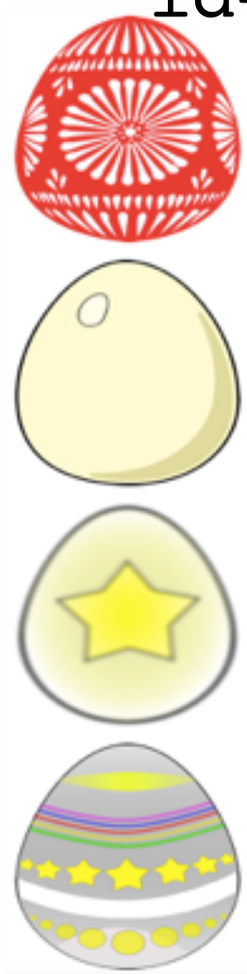
```
#egg2 {
  position:relative;
  bottom:20px;
  left:20px;
}

#egg4 {
  position:relative;
  bottom:50px;
  right:10px;
}
```

position:relative

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

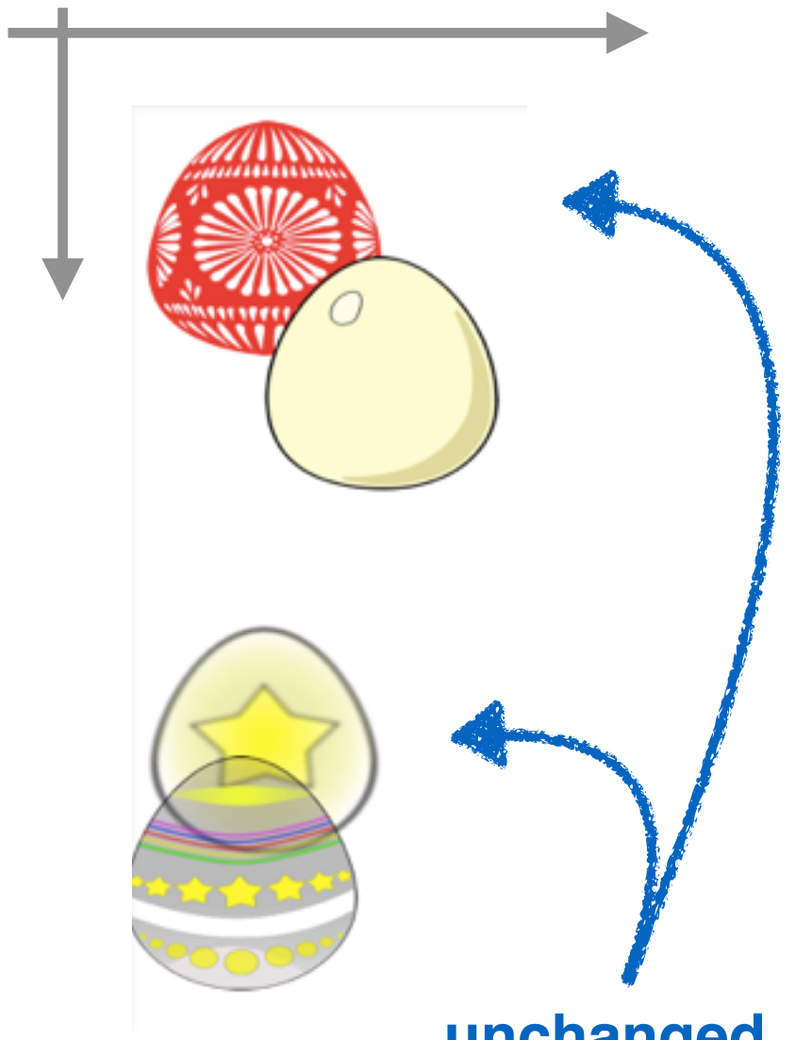
movement is **relative** to its original position

id="egg1"



id="egg4"

```
#egg2 {
  position:relative;
  bottom:20px;
  left:20px;
}

#egg4 {
  position:relative;
  bottom:50px;
  right:10px;
}
```



`position:absolute`

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

position:absolute

- the element is taken out of the normal flow (no space is reserved)
- positioning is relative to nearest ancestor or the window

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"

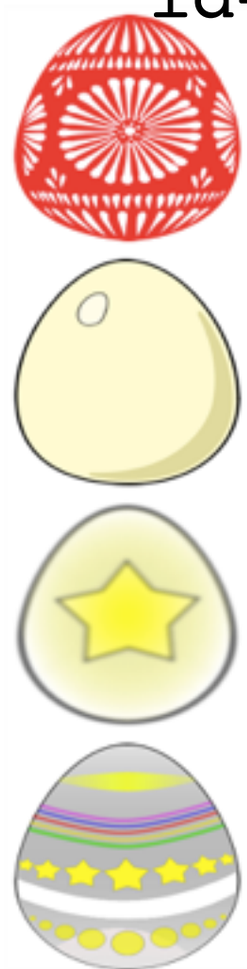
id="egg4"

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"

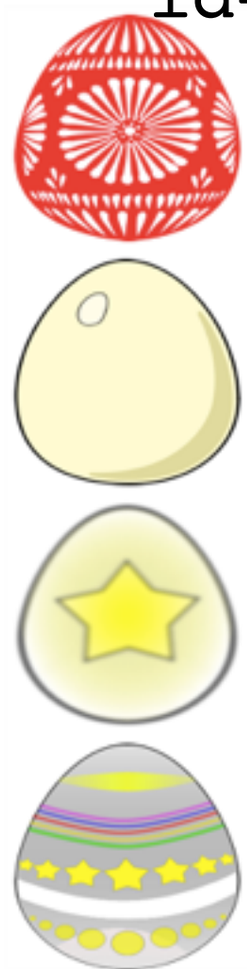
```
#egg2 {
  position:absolute;
  bottom:50px;
  left:0px;
}

#egg4 {
  position:absolute;
  bottom:0px;
  right:0px;
}
```

position:absolute

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

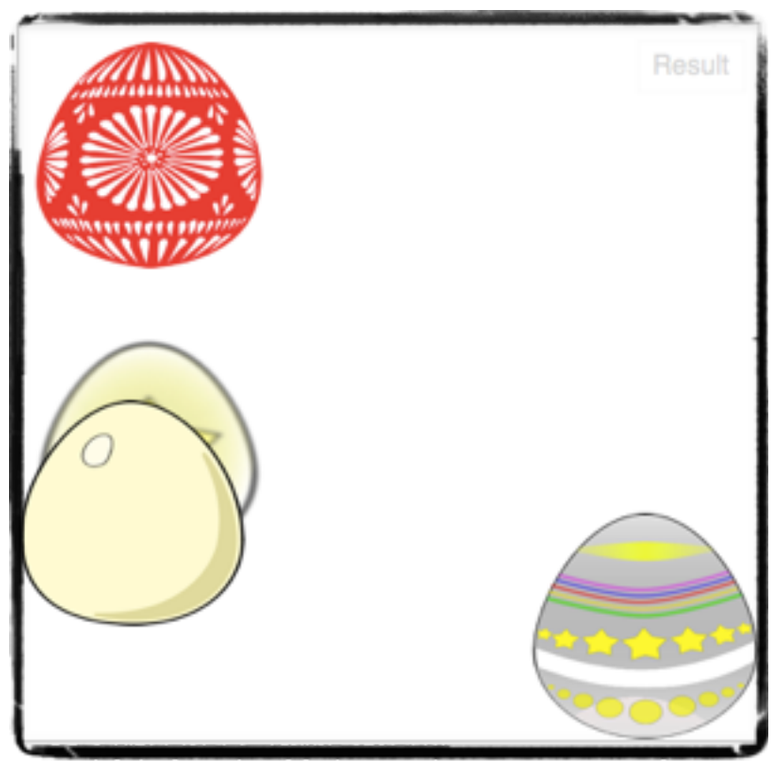
positioning is relative to nearest ancestor or the window

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 }
```



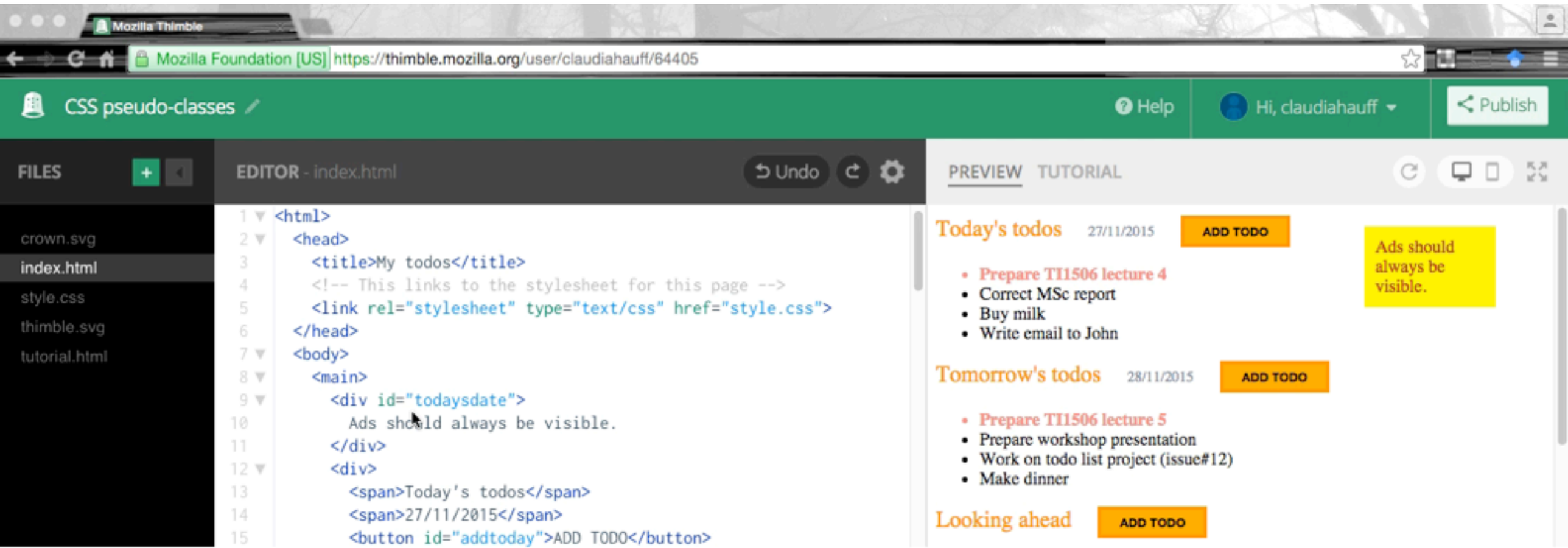
window

`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

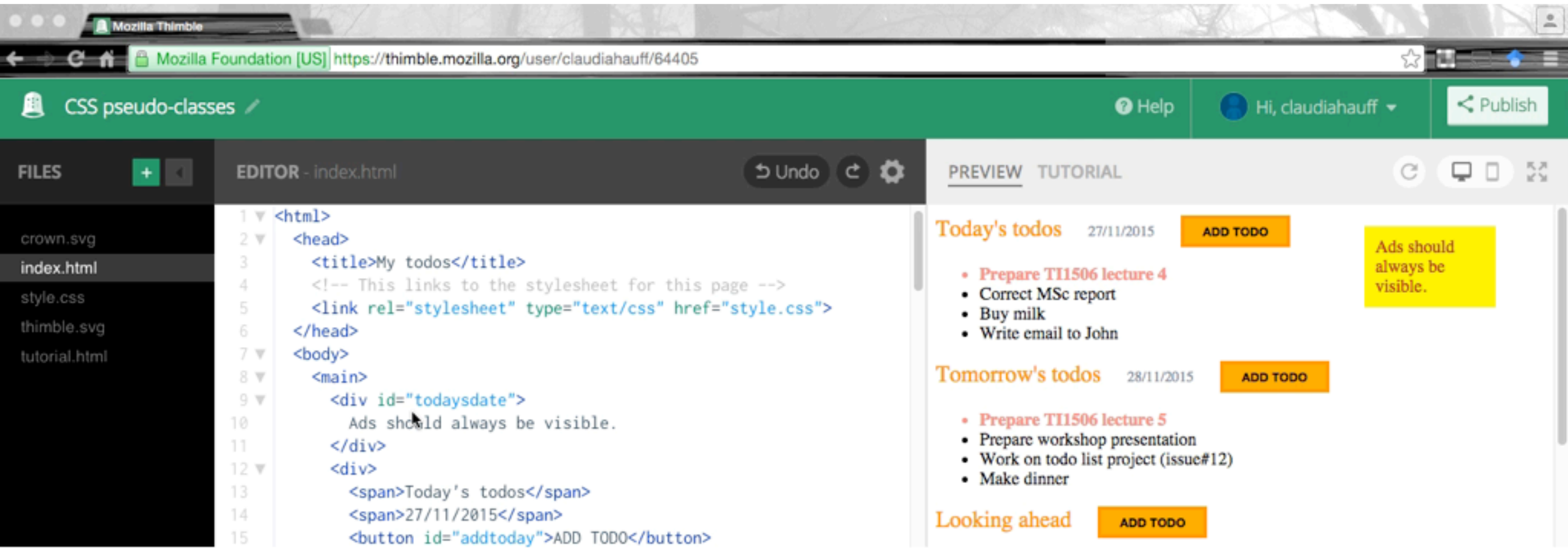


`position:fixed`

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



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

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

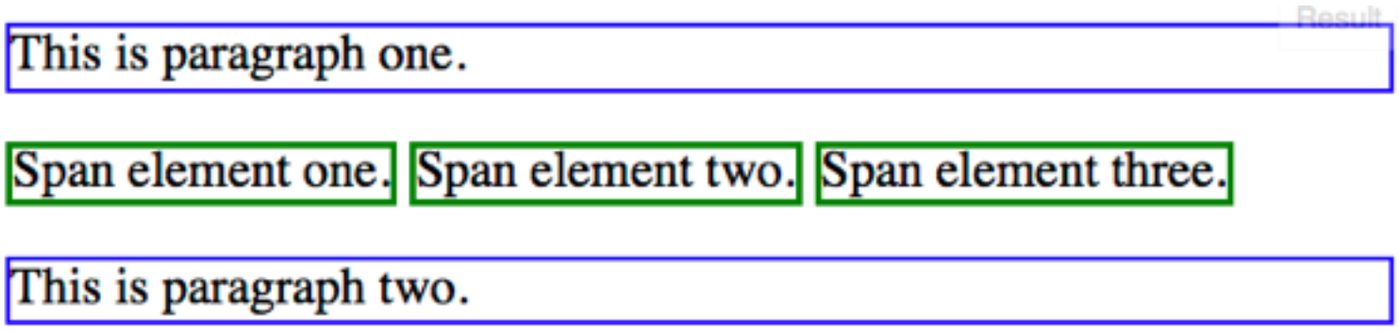
display

`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



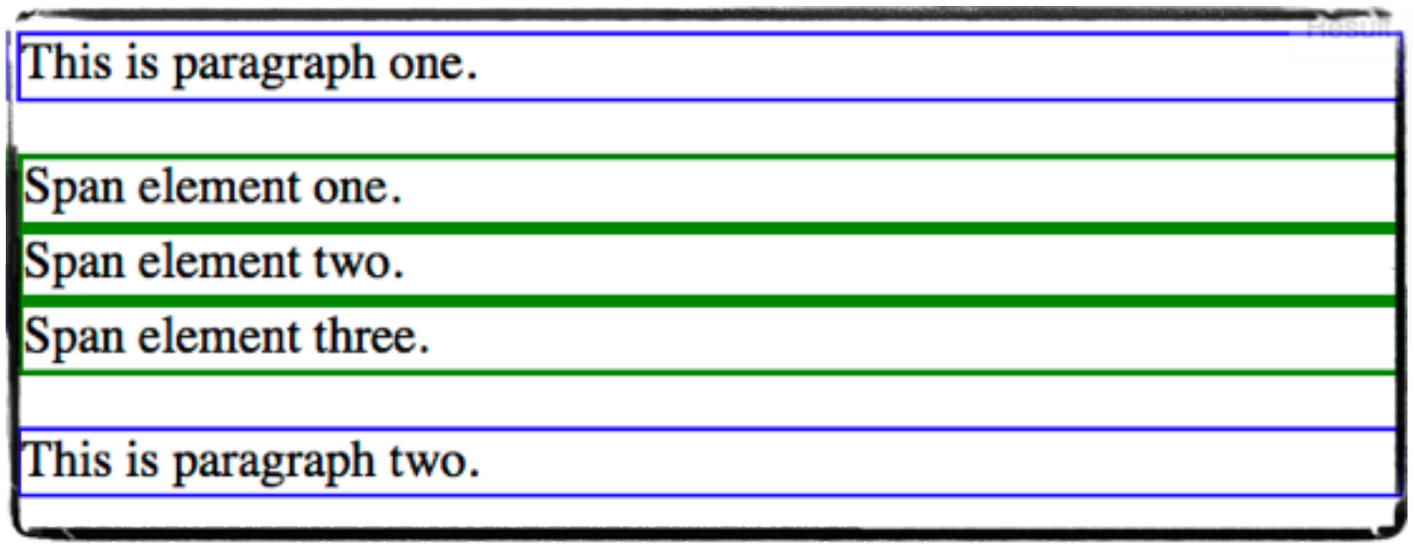
display

`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; }
```

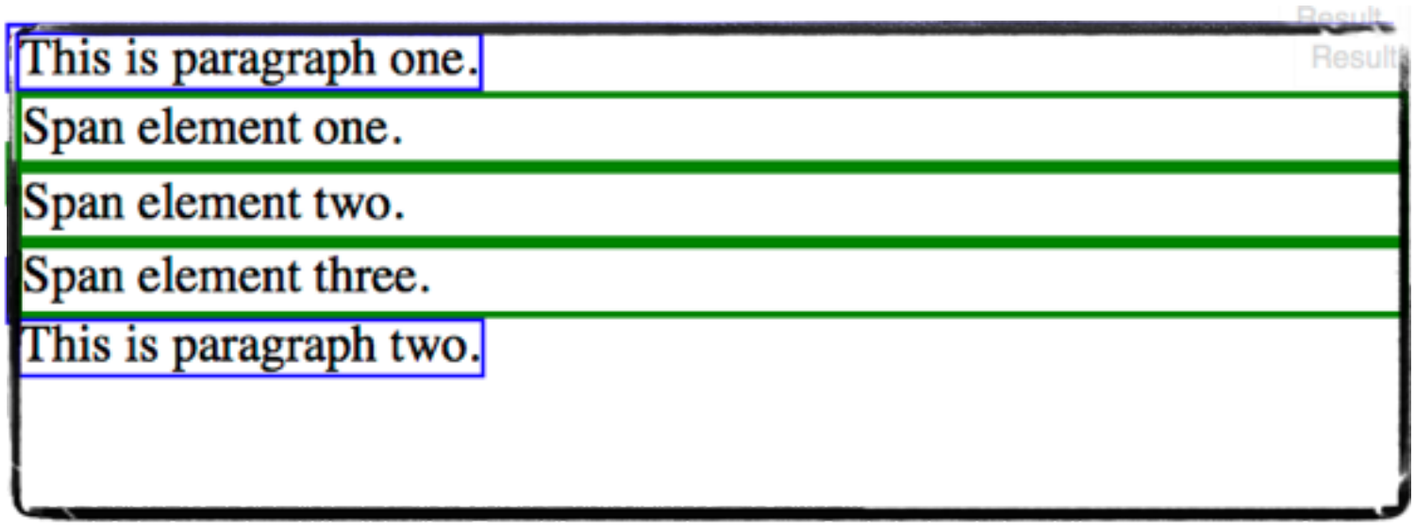
display

`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;}
```

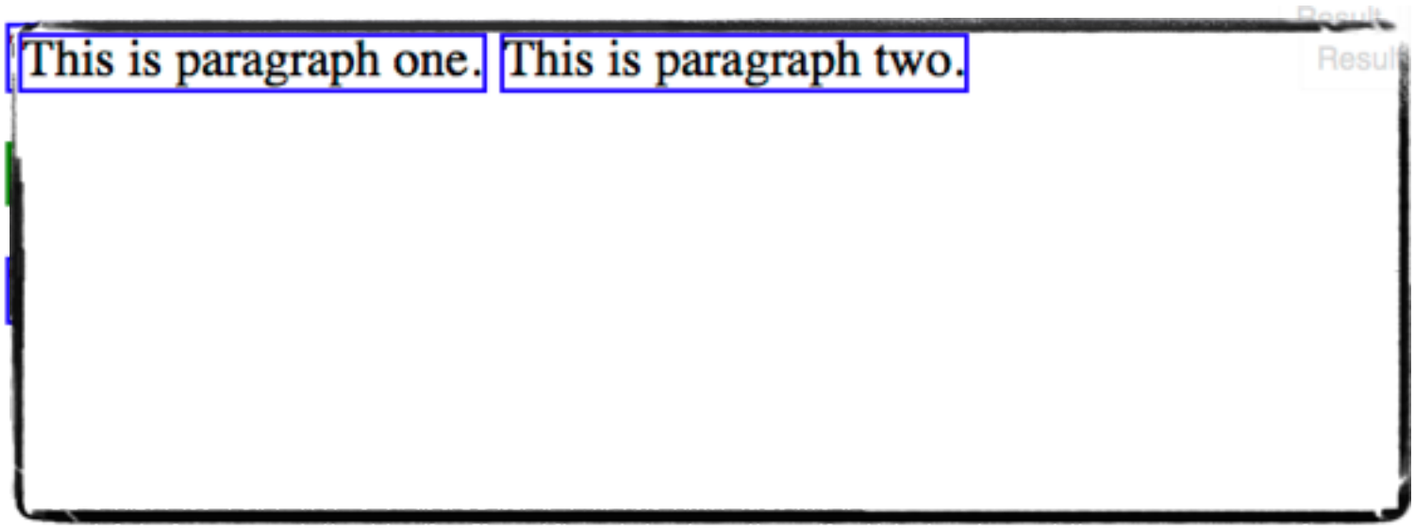
display

`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;}
```

CSS media queries

Not just one device but many…

Not just one device but many…

- Different devices should be served different styles, e.g.
  - **Printing** a todo list: ideally only b/w, no color blocks
  - **Viewing** a todo list on a **small screen**: remove non-essential information (footer, etc.)
  - **Viewing** a todo list on a **large screen**: present all available information
  - **Text-to-speech** devices: remove non-essential information (e.g. http://responsivevoice.org/)

Not just one device but many…

- Different devices should be served different styles, e.g.
  - **Printing** a todo list: ideally only b/w, no color blocks
  - **Viewing** a todo list on a **small screen**: remove non-essential information (footer, etc.)
  - **Viewing** a todo list on a **large screen**: present all available information
  - **Text-to-speech** devices: remove non-essential information (e.g. http://responsivevoice.org/)
- CSS media queries enable the use of **device-dependent** (i.e. media-type dependent) stylesheets

Not just one device but many…

- Different devices should be served different styles, e.g.
  - **Printing** a todo list: ideally only b/w, no color blocks
  - **Viewing** a todo list on a **small screen**: remove non-essential information (footer, etc.)
  - **Viewing** a todo list on a **large screen**: present all available information
  - **Text-to-speech** devices: remove non-essential information (e.g. http://responsivevoice.org/)
- CSS media queries enable the use of **device-dependent** (i.e. media-type dependent) stylesheets

HTML: write once

Not just one device but many…

- Different devices should be served different styles, e.g.
  - **Printing** a todo list: ideally only b/w, no color blocks
  - **Viewing** a todo list on a **small screen**: remove non-essential information (footer, etc.)
  - **Viewing** a todo list on a **large screen**: present all available information
  - **Text-to-speech** devices: remove non-essential information (e.g. http://responsivevoice.org/)
- CSS media queries enable the use of **device-dependent** (i.e. media-type dependent) stylesheets

HTML: write once

CSS: write once per device

Not just one device but many…

- Different devices should be served different styles, e.g.
  - **Printing** a todo list: ideally only b/w, no color blocks
  - **Viewing** a todo list on a **small screen**: remove non-essential information (footer, etc.)
  - **Viewing** a todo list on a **large screen**: present all available information
  - **Text-to-speech** devices: remove non-essential information (e.g. http://responsivevoice.org/)

HTML5 technology

- CSS media queries enable the use of **device-dependent** (i.e. media-type dependent) stylesheets

HTML: write once

CSS: write once per device

Media queries can be complex

Media types: **`all`**, **`print`**, **`screen`**, **`speech`**

```
<link rel="stylesheet"
   media="screen and (min-width: 800px),
          (min-width: 3000px)"
   href="large-device.css">
…
<style>
    @media print {
      body {
        color: black !important;
        width: 100%;
      }
    }
    @media screen and (max-width: 300px) {
      #sidebar {
        display: none;
      }
    }
</style>
```

Media queries can be complex

Media types: **`all`**, **`print`**, **`screen`**, **`speech`**

```
<link rel="stylesheet"
   media="screen and (min-width: 800px),
             (min-width: 3000px)"
   href="large-device.css">
…
<style>
     @media print {
       body {
         color: black !important;
         width: 100%;
       }
     }
     @media screen and (max-width: 300px) {
       #sidebar {
         display: none;
       }
     }
</style>
```

dedicated CSS files

Media queries can be complex

Media types: **`all`**, **`print`**, **`screen`**, **`speech`**

```
<link rel="stylesheet"
   media="screen and (min-width: 800px),
          (min-width: 3000px)"
   href="large-device.css">
…
<style>
     @media print {
       body {
         color: black !important;
         width: 100%;
       }
     }
     @media screen and (max-width: 300px) {
       #sidebar {
         display: none;
       }
     }
</style>
```

dedicated CSS files

rules for different devices in one file

Media queries can be complex

Media types: **`all`**, **`print`**, **`screen`**, **`speech`**

```
<link rel="stylesheet"
  media="screen and (min-width: 800px),
         (min-width: 3000px)"
  href="large-device.css">
…
<style>
    @media print {
      body {
        color: black !importa
        width: 100%;
      }
    }
    @media screen and (max-width: 300px) {
      #sidebar {
        display: none;
      }
    }
</style>
```

dedicated CSS files

rules for different devices in one file

when printing, use black and white

Media queries can be complex

Media types: **`all`**, **`print`**, **`screen`**, **`speech`**

```
1  <link rel="stylesheet"
2    media="screen and (min-width: 800px),
3              (min-width: 3000px)"
4    href="large-device.css">
5  …
6  <style>
7      @media print {
8        body {
9          color: black !importa
10         width: 100%;
11       }
12     }
13     @media screen and (max-width: 300px) {
14       #sidebar {
15         display: none;
16       }
17     }
18 </style>
```

dedicated CSS files

rules for different devices in one file

when printing, use black and white

hide the sidebar for small devices

Media queries can be complex

Media types: **`all`**, **`print`**, **`screen`**, **`speech`**

```
1  <link rel="stylesheet"
2    media="screen and (min-width: 800px),
3              (min-width: 3000px)"
4    href="large-device.css">
5  …
6  <style>
7      @media print {
8        body {
9          color: black !importa
10         width: 100%;
11       }
12     }
13     @media screen and (max-width: 300px) {
14       #sidebar {
15         display: none;
16       }
17     }
18 </style>
```

dedicated CSS files

rules for different devices in one file

when printing, use black and white

"and": logical and

hide the sidebar for small devices

Media queries can be complex

Media types: **`all`**, **`print`**, **`screen`**, **`speech`**

```
1  <link rel="stylesheet"
2    media="screen and (min-width: 800px),
3              (min-width: 3000px)"
4    href="large-device.css">
5  …
6  <style>
7      @media print {
8        body {
9          color: black !importa
10         width: 100%;
11       }
12     }
13     @media screen and (max-width: 300px) {
14       #sidebar {
15         display: none;
16       }
17     }
18 </style>
```

“,”: logical or

dedicated CSS files

rules for different devices in one file

when printing, use black and white

“and”: logical and

hide the sidebar for small devices

Animations and transitions

QuickTime Player File Edit View Window Help 85% Fri 06:27 Claudia Hauff

Mozilla Thimble | ResponsiveVoice.JS

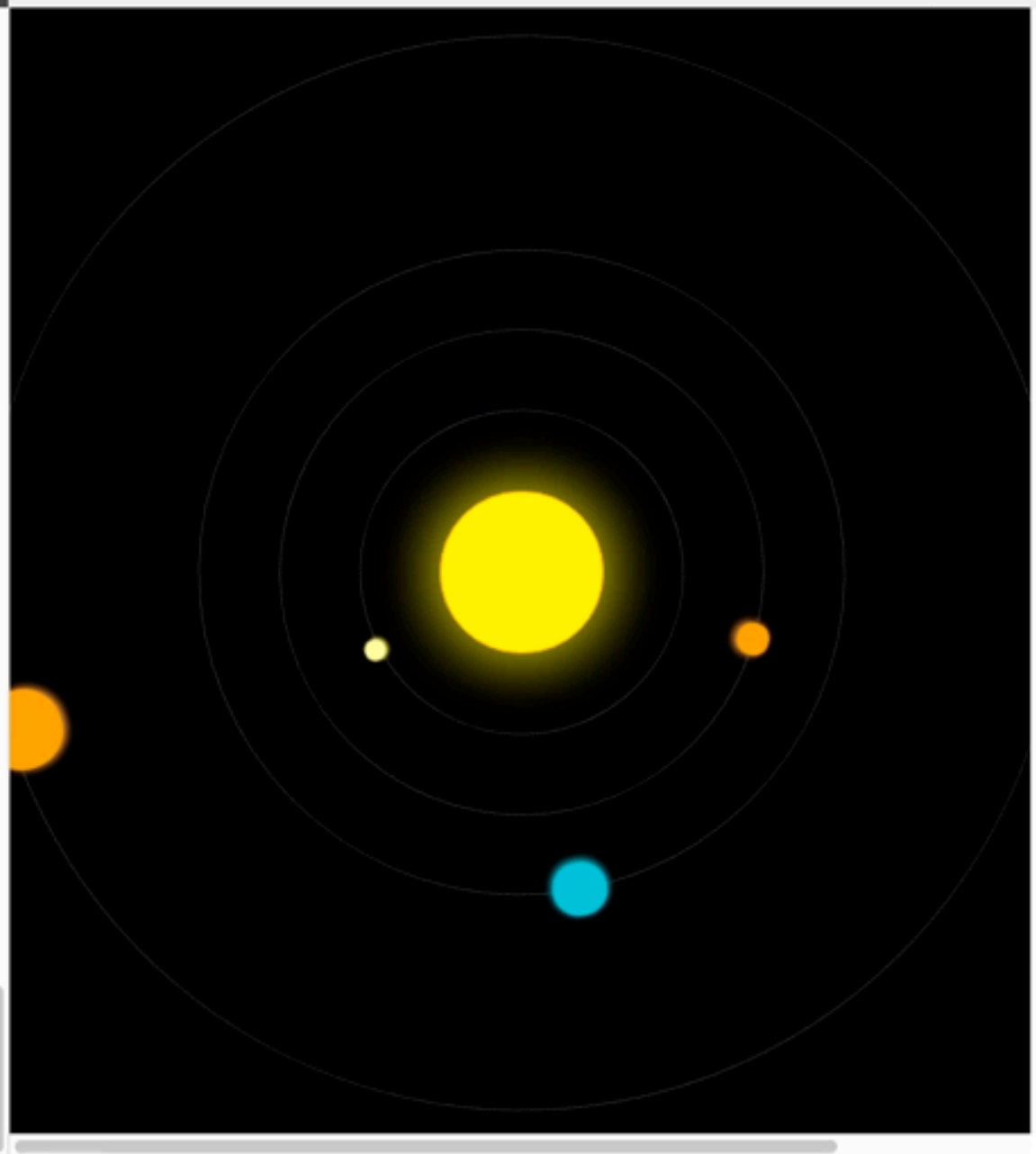
Mozilla Foundation [US] https://thimble.mozilla.org/user/claudiahauff/64593

Animations | Help | Hi, claudiahauff | Publish

FILES | EDITOR - style.css | Undo | PREVIEW

index.html
style.css

```
188   border-color: #333;
189   border-radius: 50%;
190
191   /* ... your other orbit styles ... */
192
193   -webkit-animation: spin-left 35s linear infinite;
194   -moz-animation: spin-left 35s linear infinite;
195   -ms-animation: spin-left 35s linear infinite;
196   -o-animation: spin-left 35s linear infinite;
197   animation: spin-left 35s linear infinite;
198 }
199
200
201 @-webkit-keyframes spin-left {
202   100% {
203     -webkit-transform: rotate(-360deg);
204     -moz-transform: rotate(-360deg);
205     -ms-transform: rotate(-360deg);
206     -o-transform: rotate(-360deg);
207     transform: rotate(-360deg);
208   }
209 }
210
211 @keyframes spin-left {
212   100% {
213     -webkit-transform: rotate(-360deg);
214     -moz-transform: rotate(-360deg);
215     -ms-transform: rotate(-360deg);
216     -o-transform: rotate(-360deg);
217     transform: rotate(-360deg);
218   }
219 }
220
```



QuickTime Player File Edit View Window Help 85% Fri 06:27 Claudia Hauff

Mozilla Thimble | ResponsiveVoice.JS

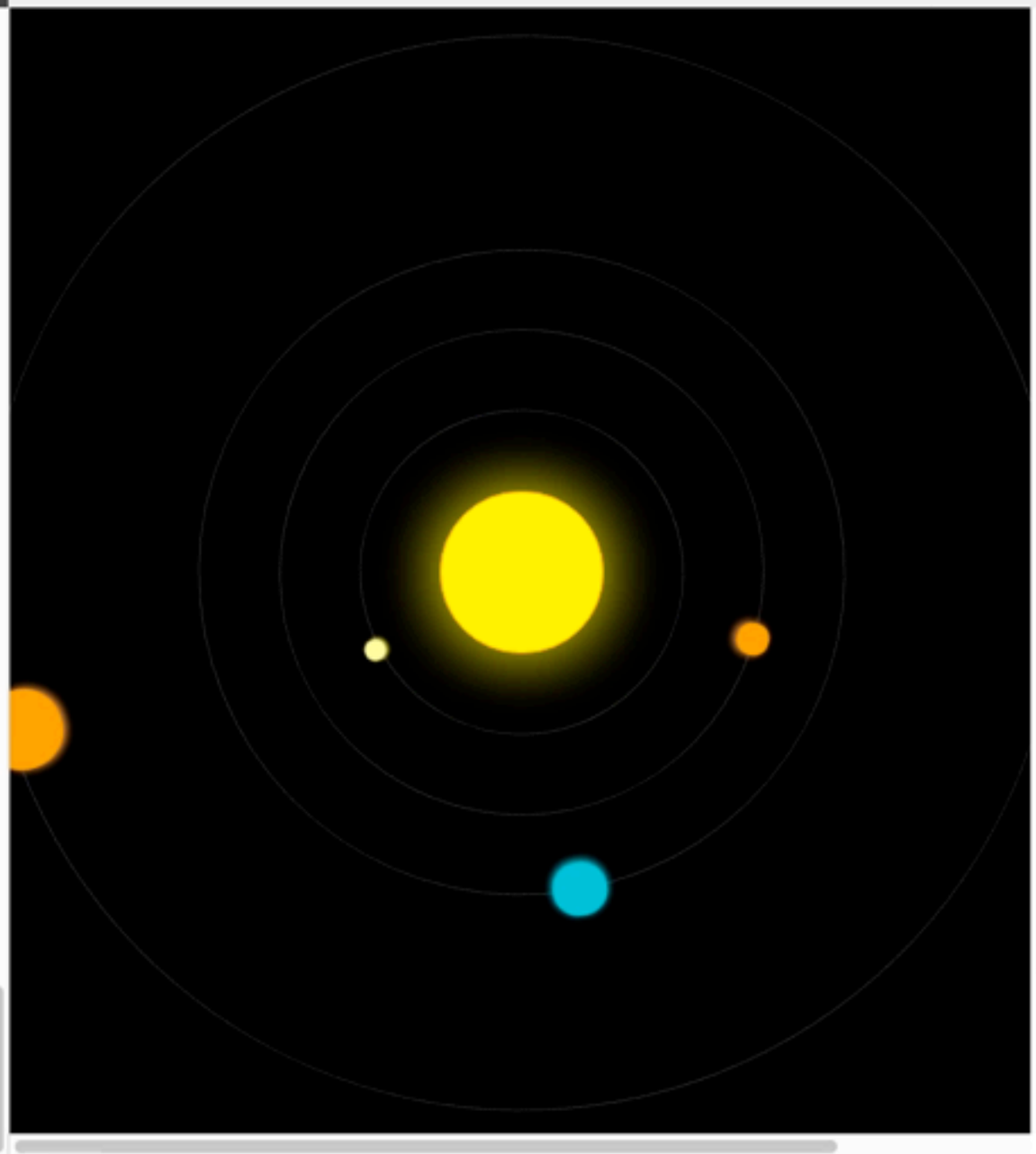
Mozilla Foundation [US] https://thimble.mozilla.org/user/claudiahauff/64593

Animations | Help | Hi, claudiahauff | Publish

FILES | EDITOR - style.css | Undo | PREVIEW

index.html
style.css

```
188   border-color: #333;
189   border-radius: 50%;
190
191   /* ... your other orbit styles ... */
192
193   -webkit-animation: spin-left 35s linear infinite;
194   -moz-animation: spin-left 35s linear infinite;
195   -ms-animation: spin-left 35s linear infinite;
196   -o-animation: spin-left 35s linear infinite;
197   animation: spin-left 35s linear infinite;
198 }
199
200
201 @-webkit-keyframes spin-left {
202   100% {
203     -webkit-transform: rotate(-360deg);
204     -moz-transform: rotate(-360deg);
205     -ms-transform: rotate(-360deg);
206     -o-transform: rotate(-360deg);
207     transform: rotate(-360deg);
208   }
209 }
210
211 @keyframes spin-left {
212   100% {
213     -webkit-transform: rotate(-360deg);
214     -moz-transform: rotate(-360deg);
215     -ms-transform: rotate(-360deg);
216     -o-transform: rotate(-360deg);
217     transform: rotate(-360deg);
218   }
219 }
220
```



In general …

In general …

- CSS styles (states) are defined by the user, the rendering engine takes care of the transition between styles

In general …

- CSS styles (states) are defined by the user, the rendering engine takes care of the transition between styles

- **Animations** consist of:
  - an animation style (linear, etc.)
  - a number of “keyframes” that act as transition waypoints

In general …

- CSS styles (states) are defined by the user, the rendering engine takes care of the transition between styles

- **Animations** consist of:
  - an animation style (linear, etc.)
  - a number of “keyframes” that act as transition waypoints

- **Transitions** are animations (with a simpler syntax):
  - that consist of exactly 2 states: start and end state

CSS vs. JavaScript animations

CSS vs. JavaScript animations

- **Easy to use** (standard CSS) — no need to learn JavaScript

CSS vs. JavaScript animations

- **Easy to use** (standard CSS) — no need to learn JavaScript
- Rendering engines are **optimised** for CSS-based animations

CSS vs. JavaScript animations

- **Easy to use** (standard CSS) — no need to learn JavaScript

- Rendering engines are **optimised** for CSS-based animations

- CSS animations can do much more than animating buttons

CSS animation example (Firefox)

```
1  #p1 {
2      animation-duration: 5s;
3      animation-name: pToRight;
4      top: 5px; left: 5px;
5  }
6
7  @keyframes pToRight {
8      from {
9          top:5px; left:5px;
10         background-color: lightgreen;
11     }
12     50% {
13         background-color: red;
14     }
15     to {
16         top:5px; left:250px;
17         background-color: lightblue;
18     }
19 }
```

duration of animation (seconds)

animation name (@keyframes)

http://jsfiddle.net/Lyh5qvfo/1/

CSS animation example (Firefox)

```
1  #p1 {
2      animation-duration: 5s;
3      animation-name: pToRight;
4      top: 5px; left: 5px;
5  }
6
7  @keyframes pToRight {
8      from {
9          top:5px; left:5px;
10         background-color: lightgreen;
11     }
12     50% {
13         background-color: red;
14     }
15     to {
16         top:5px; left:250px;
17         background-color: lightblue;
18     }
19 }
```

duration of animation (seconds)

animation name (@keyframes)

start state

http://jsfiddle.net/Lyh5qvfo/1/

CSS animation example (Firefox)

```
1  #p1 {
2      animation-duration: 5s;
3      animation-name: pToRight;
4      top: 5px; left: 5px;
5  }
6
7  @keyframes pToRight {
8      from {
9          top:5px; left:5px;
10         background-color: lightgreen;
11     }
12     50% {
13         background-color: red;
14     }
15     to {
16         top:5px; left:250px;
17         background-color: lightblue;
18     }
19 }
```

duration of animation (seconds)

animation name (@keyframes)

start state

intermediate state

http://jsfiddle.net/Lyh5qvfo/1/

CSS animation example (Firefox)

```
1  #p1 {
2      animation-duration: 5s;
3      animation-name: pToRight;
4      top: 5px; left: 5px;
5  }
6
7  @keyframes pToRight {
8      from {
9          top:5px; left:5px;
10         background-color: lightgreen;
11     }
12     50% {
13         background-color: red;
14     }
15     to {
16         top:5px; left:250px;
17         background-color: lightblue;
18     }
19 }
```

duration of animation (seconds)

animation name (@keyframes)

start state

intermediate state

end state

http://jsfiddle.net/Lyh5qvfo/1/

CSS animation example (-webkit-)

```
1  #p1 {
2      -webkit-animation-duration: 5s;
3      -webkit-animation-name: pToRight;
4      top: 5px; left: 5px;
5  }
6
7  @-webkit-keyframes pToRight {
8      from {
9          top:5px; left:5px;
10         background-color: lightgreen;
11     }
12     50% {
13         background-color: red;
14     }
15     to {
16         top:5px; left:250px;
17         background-color: lightblue;
18     }
19 }
```

to support different browsers, the code needs to be **repeated** for **every browser prefix**

CSS animation control

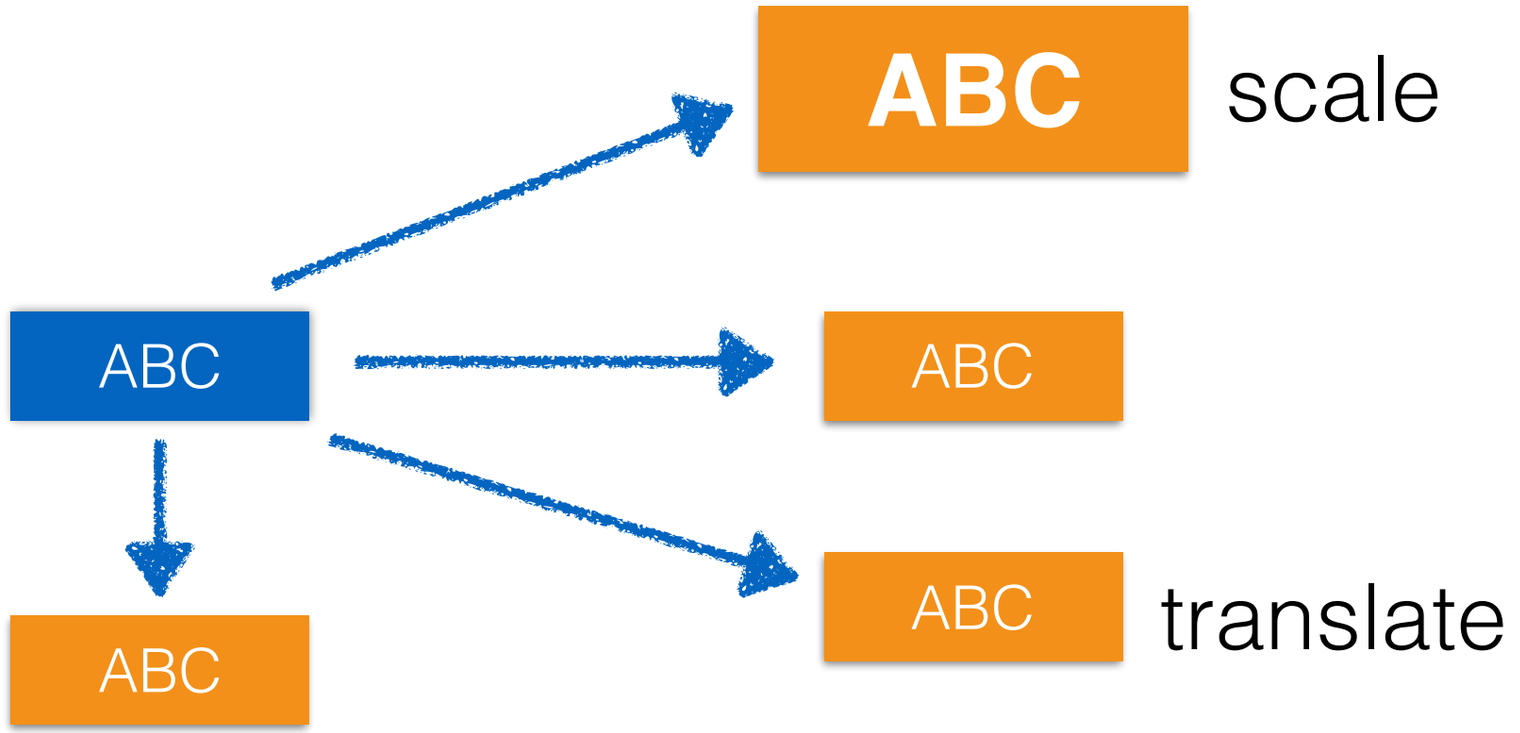
`animation-iteration-count`
number of times an animation is executed (default: 1); value either a positive number or **`infinite`**

`animation-direction`
by default the animation restarts at the starting keyframe; if set to **`alternate`** the animation direction change every iteration

`animation-delay`
number of seconds until the animation starts (default 0s)

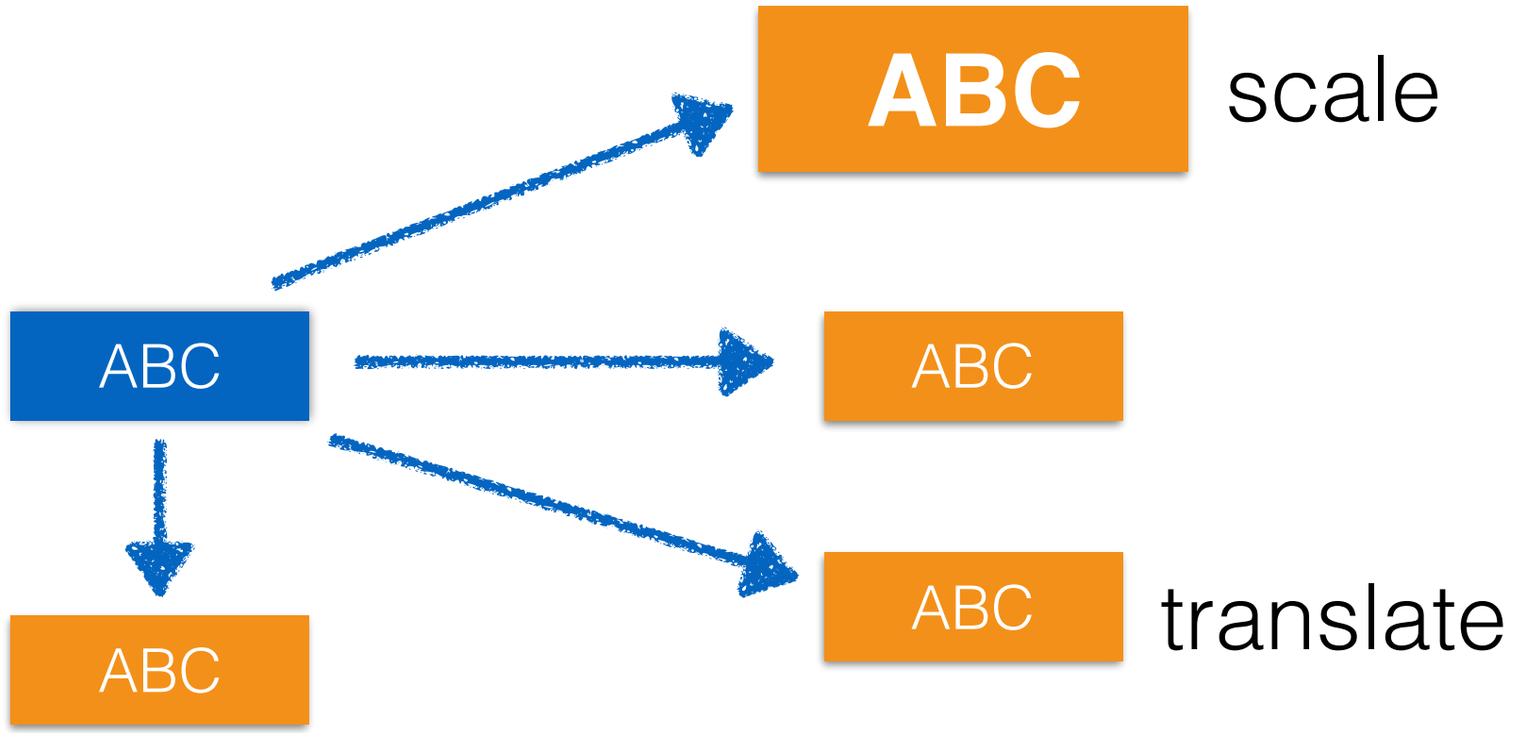
transform

So far we can:



`transform`

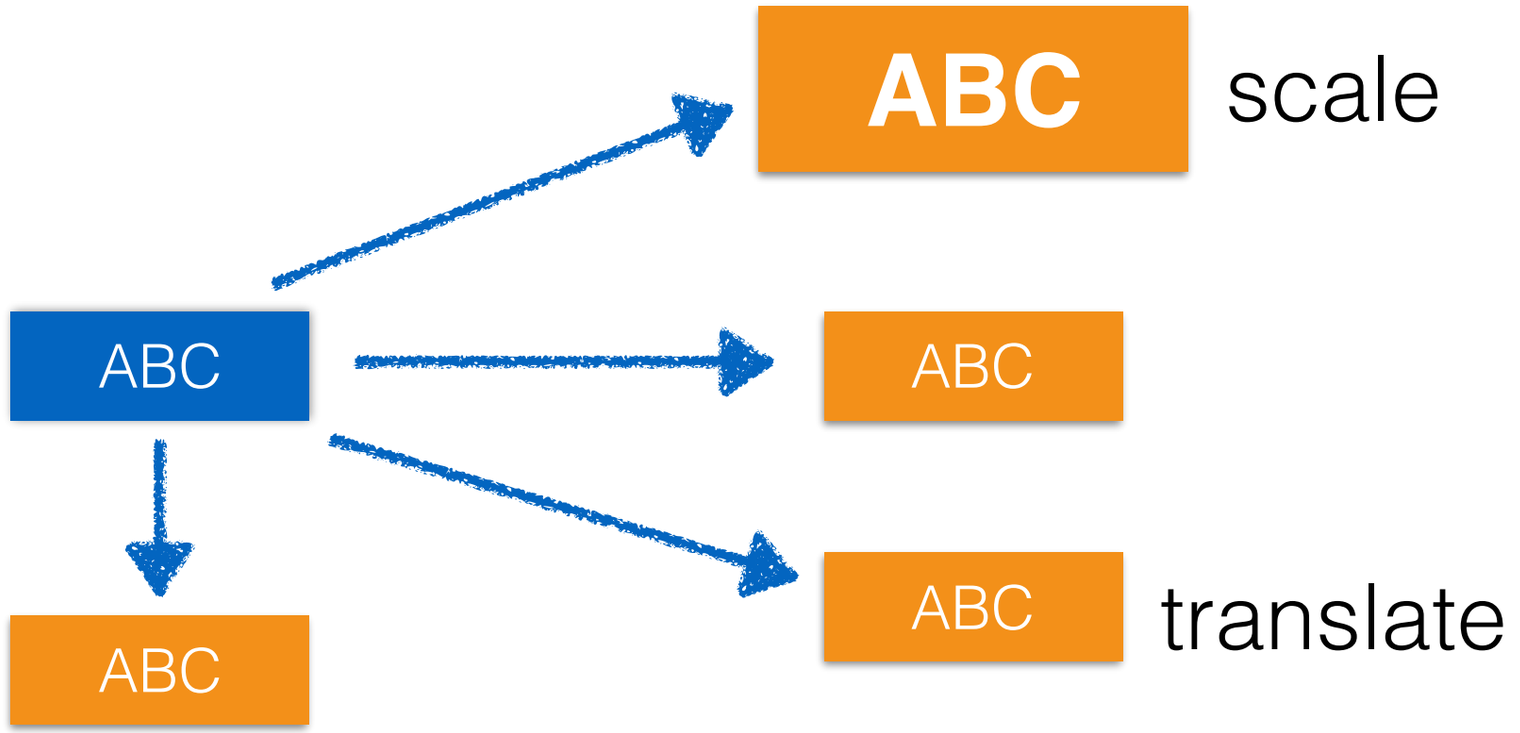
So far we can:



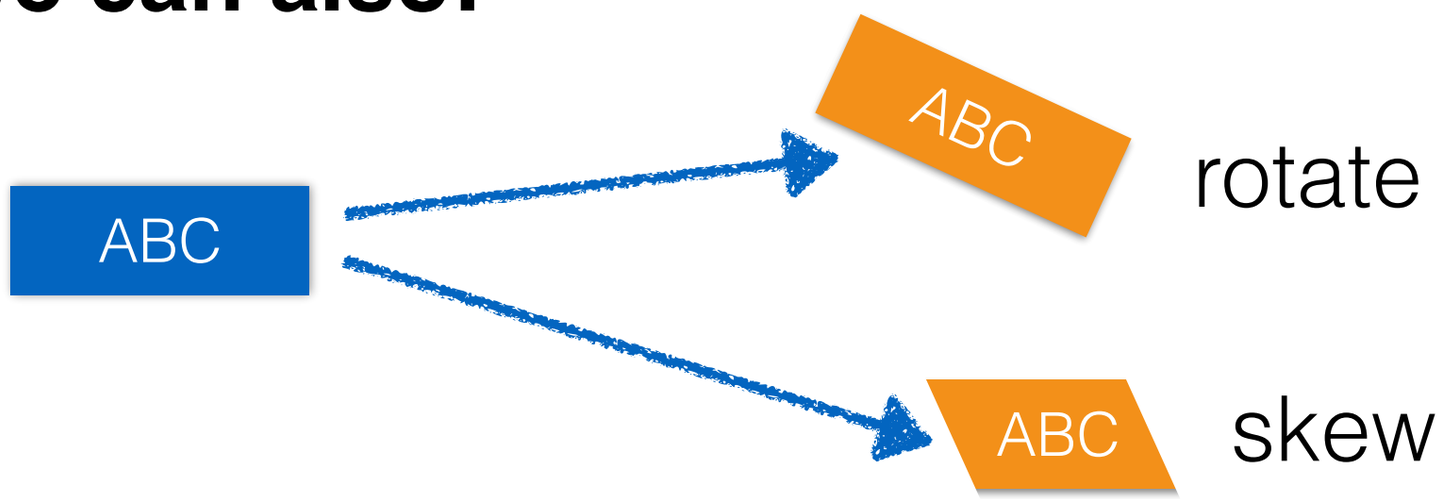
With `transform` we can also:

`transform`

So far we can:

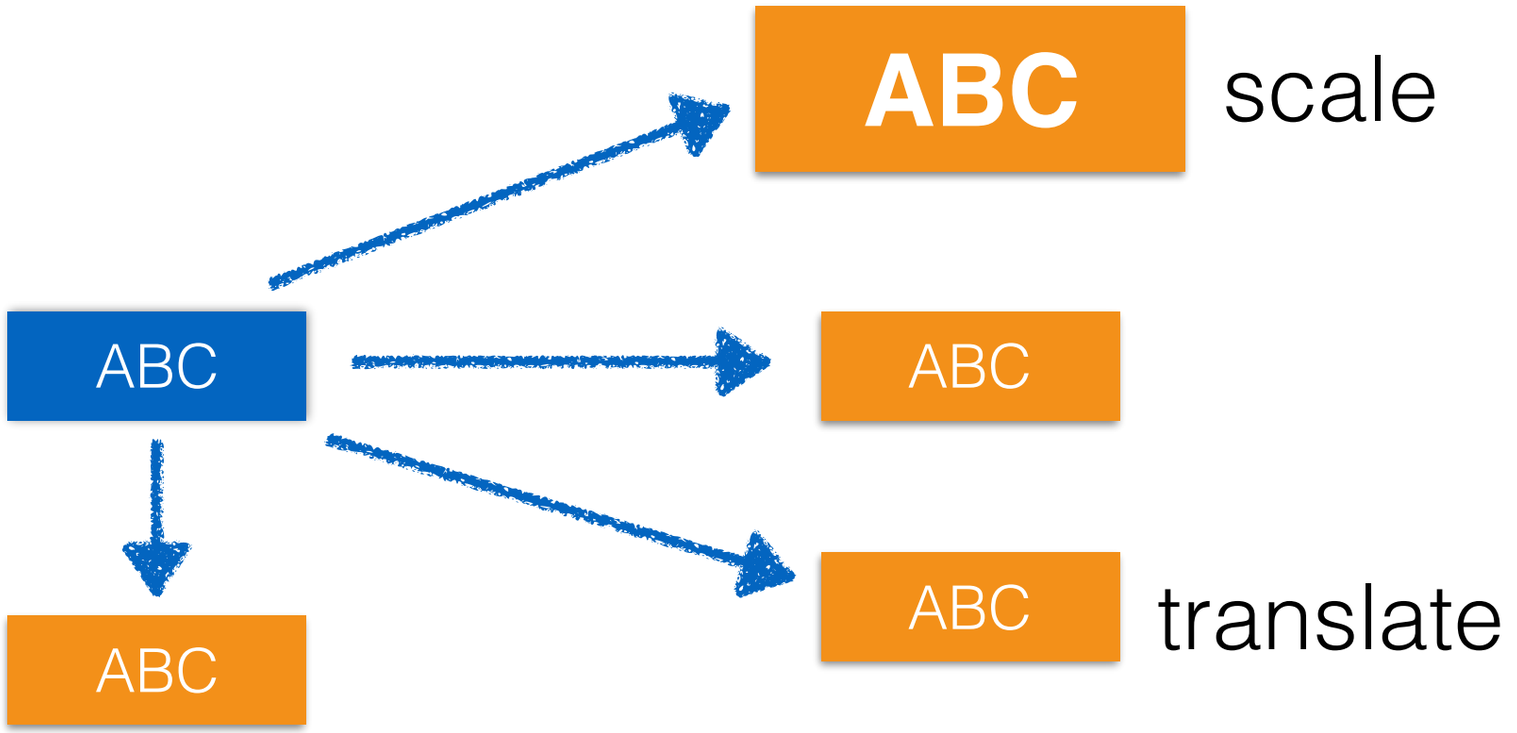


With `transform` we can also:

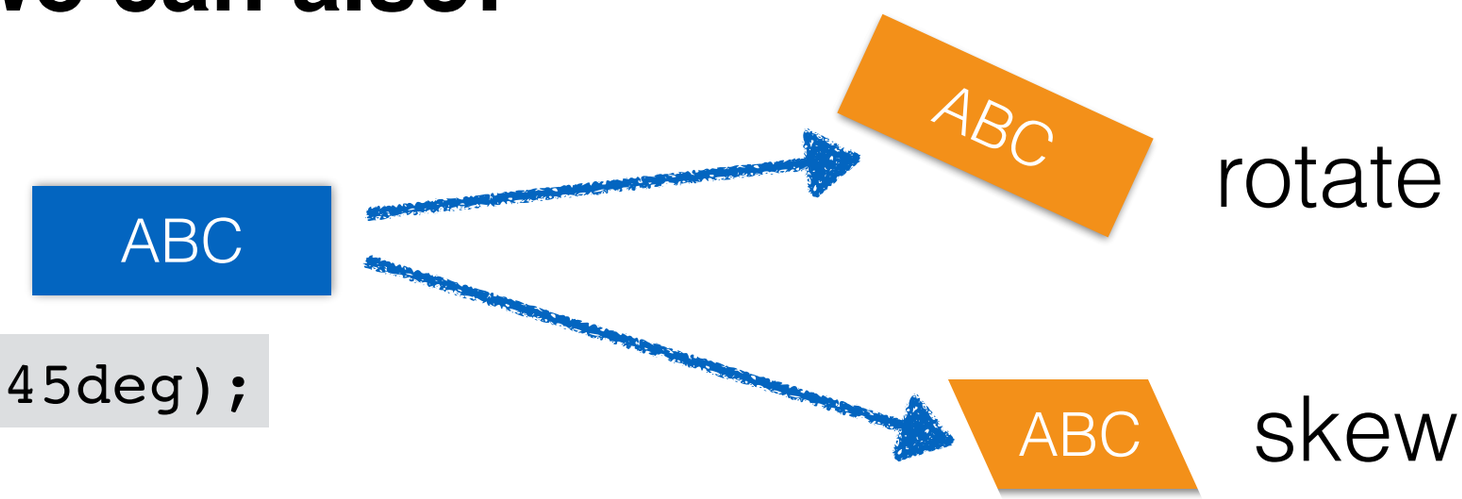


`transform`

So far we can:



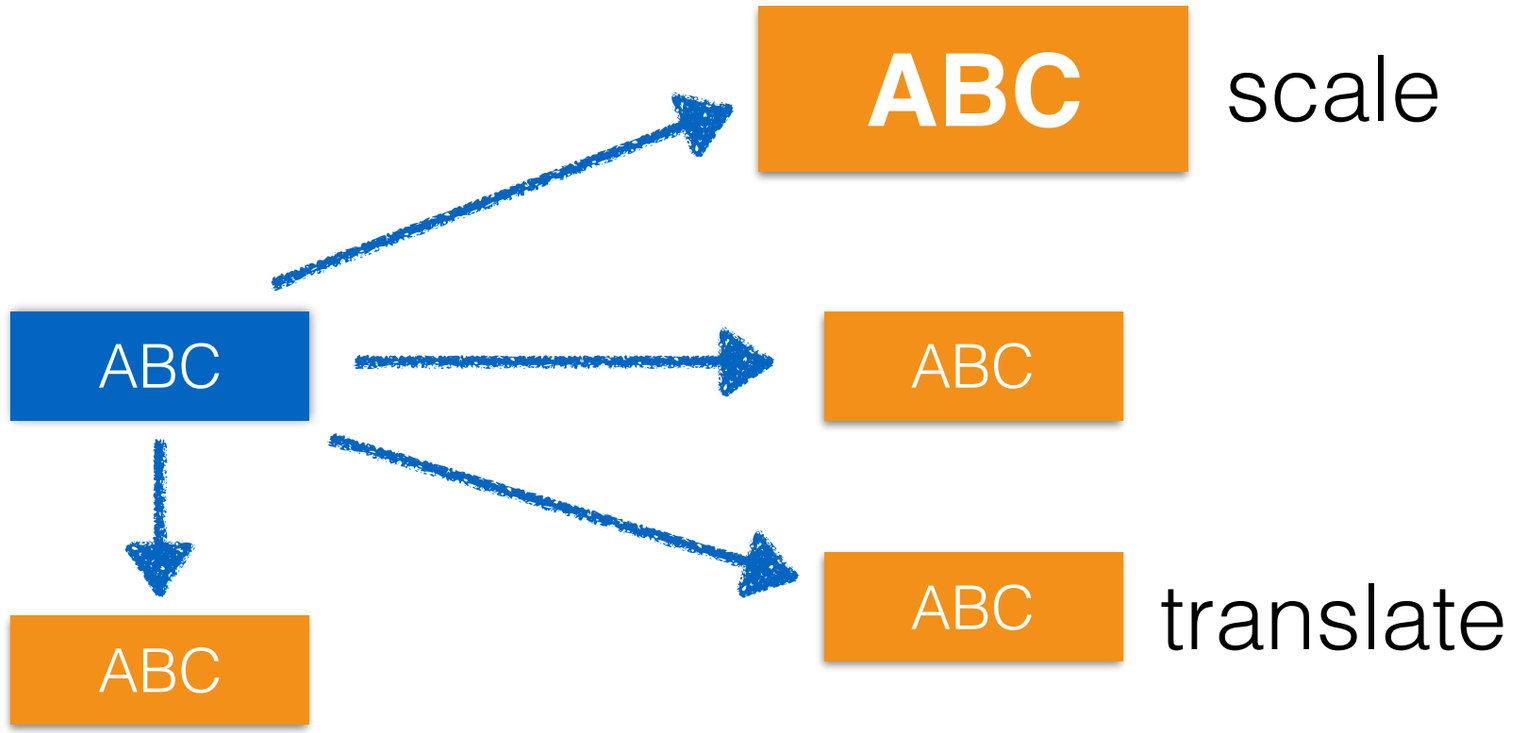
With `transform` we can also:



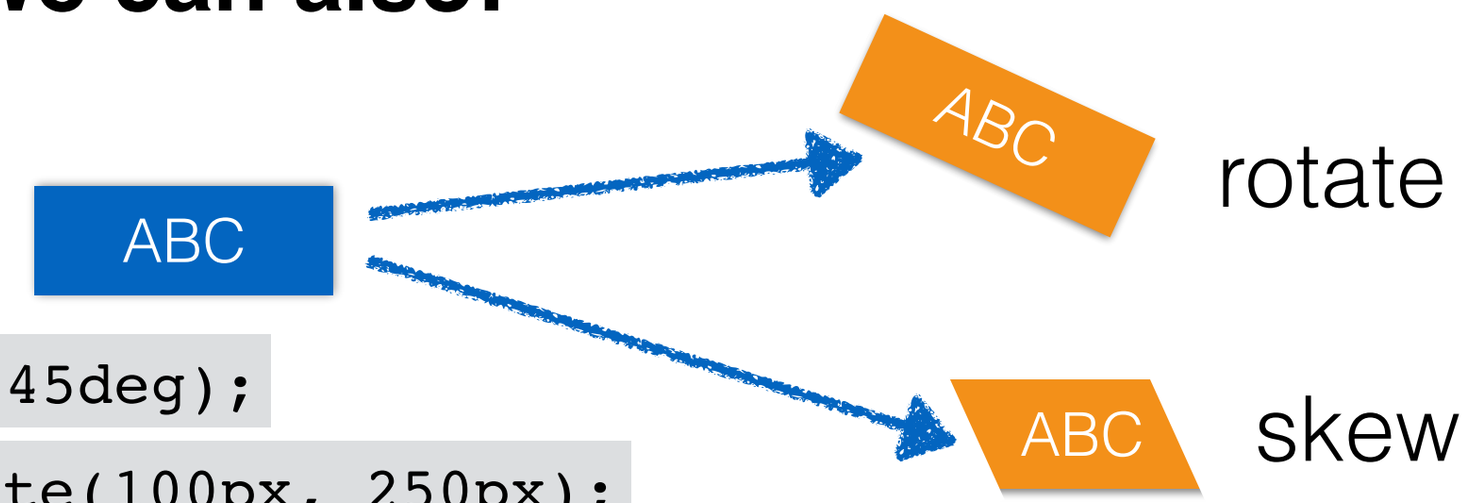

```
1 transform: rotate(45deg);
```

`transform`

So far we can:



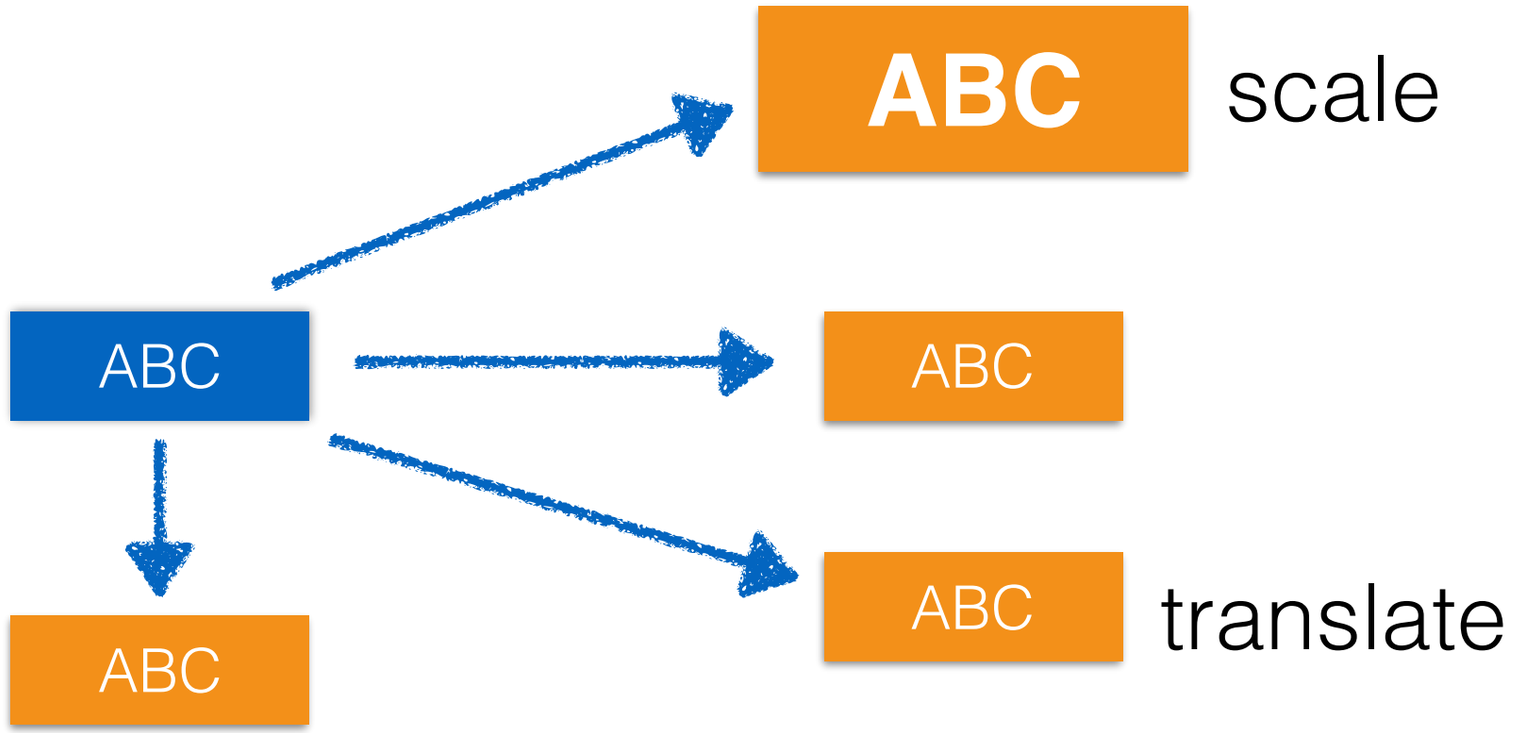
With `transform` we can also:



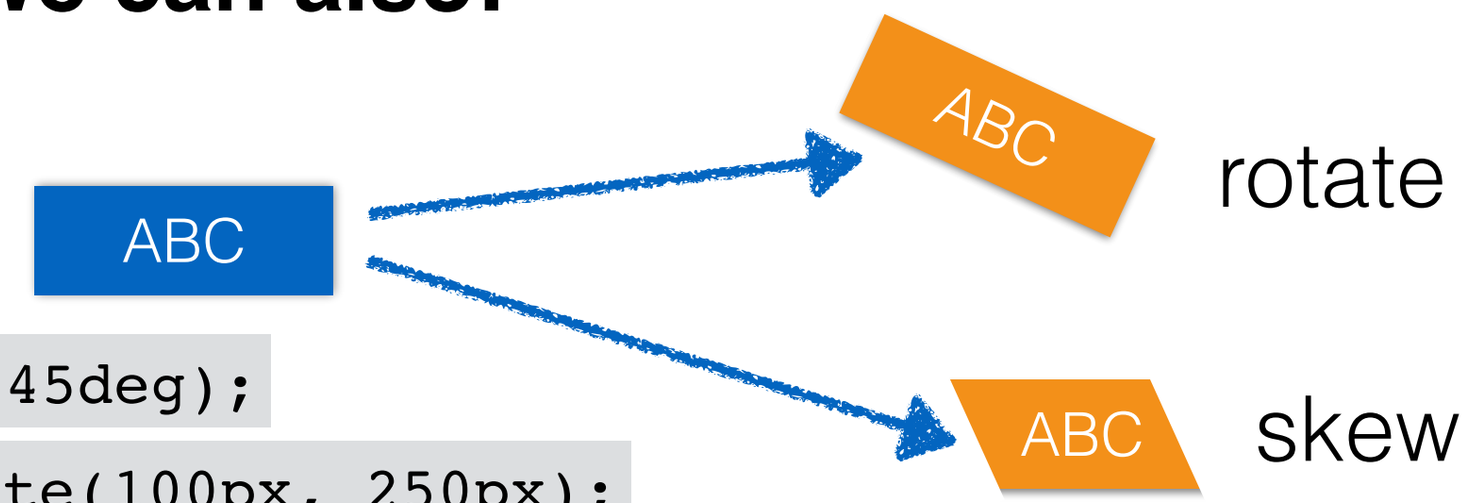

```
1 transform: rotate(45deg);
2 transform: translate(100px, 250px);
```

transform

So far we can:



With `transform` we can also:




```
1 transform: rotate(45deg);
2 transform: translate(100px, 250px);
3 transform: translate(10px,10px) skew(20deg);
```

CSS transitions

```
1  .box {
2      border-style: solid;
3      border-width: 1px;
4      display: block;
5      width: 100px;
6      height: 100px;
7      background-color: red;
8
9
10 }
11 .box:hover {
12     background-color: green;
13     width:200px;
14     height:200px;
15     -webkit-transform:rotate(180deg);
16     transform:rotate(180deg);
17 }
```

We have been using (default) transitions all the time.

CSS transitions

```
1  .box {
2      border-style: solid;
3      border-width: 1px;
4      display: block;
5      width: 100px;
6      height: 100px;
7      background-color: red;
8
9
10 }
11 .box:hover {
12     background-color: green;
13     width:200px;
14     height:200px;
15     -webkit-transform:rotate(180deg);
16     transform:rotate(180deg);
17 }
```

start state

We have been using (default) transitions all the time.

CSS transitions

```
1  .box {
2      border-style: solid;
3      border-width: 1px;
4      display: block;
5      width: 100px;
6      height: 100px;
7      background-color: red;
8
9
10 }
11 .box:hover {
12     background-color: green;
13     width:200px;
14     height:200px;
15     -webkit-transform:rotate(180deg);
16     transform:rotate(180deg);
17 }
```

start state

end state

We have been using (default) transitions all the time.

CSS transitions

```
1  .box {
2      border-style: solid;
3      border-width: 1px;
4      display: block;
5      width: 100px;
6      height: 100px;
7      background-color: red;
8      -webkit-transition:width 2s, height 2s, -webkit-transform 2s;
9      transition:width 2s, height 2s, transform 2s;
10 }
11 .box:hover {
12      background-color: green;
13      width:200px;
14      height:200px;
15      -webkit-transform:rotate(180deg);
16      transform:rotate(180deg);
17 }
```

start state

declare transition

end state

We have been using (default) transitions all the time.

Ensure the IMDB database is available in your VM.

Needed in Monday's lecture.

flickr@courtneyrian