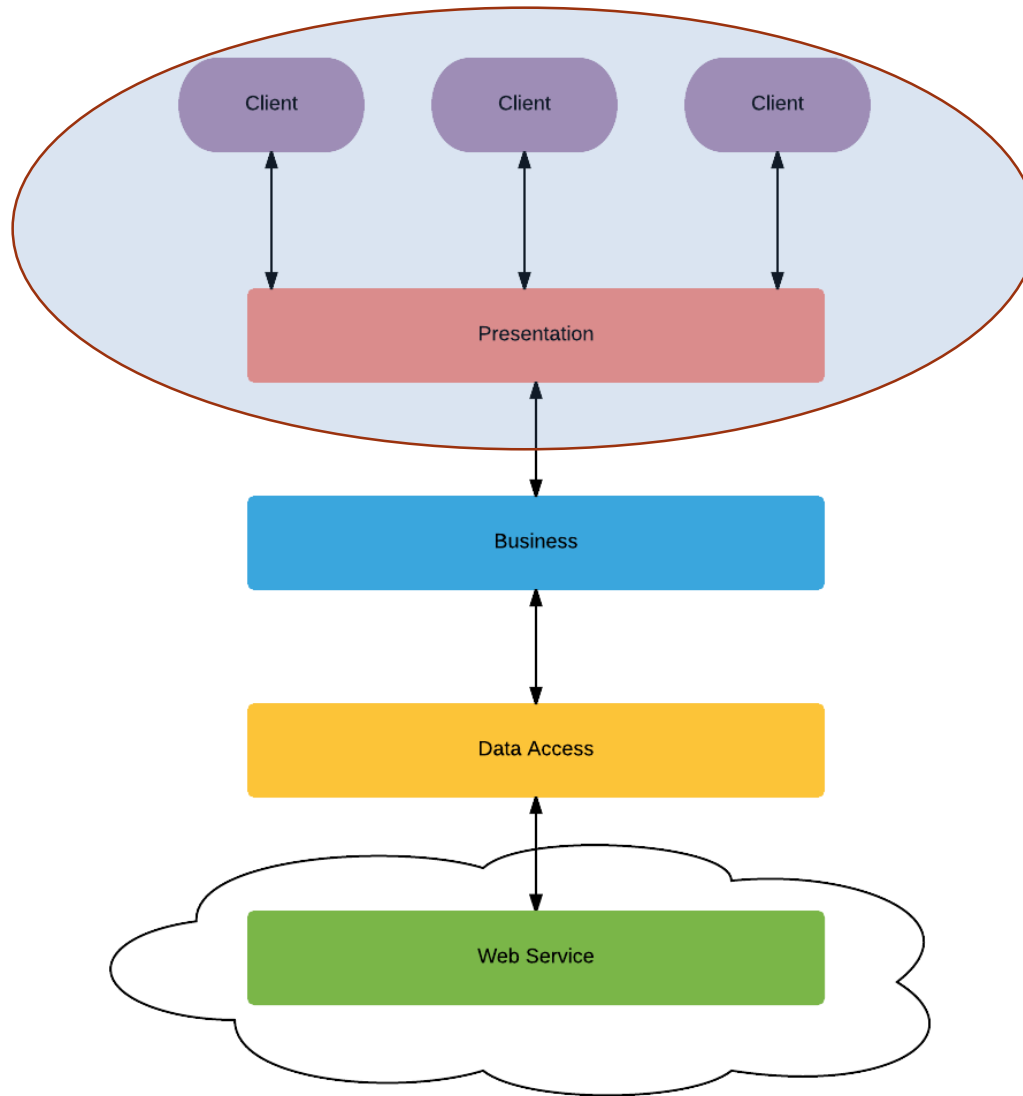


RAWDATA SECTION 4

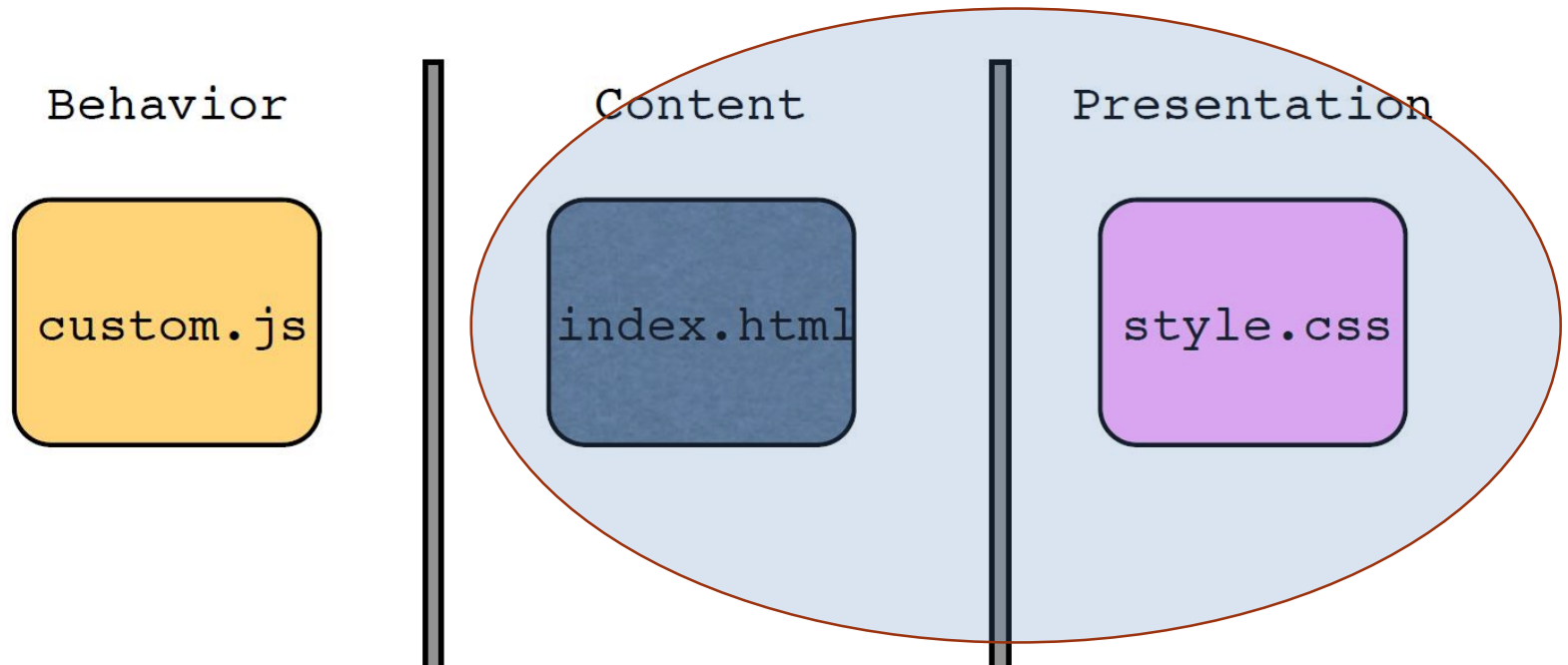
Troels Andreassen & Henrik Bulskov



SYSTEM DEVELOPMENT



UNOBTRUSIVE DESIGN



KNOCKOUT

- Bindings
- Observables
- Templates and Control of Flow



Dependency
Tracking
via Observables

Declarative
bindings

Templating
Support

KEY KNOCKOUT CONCEPTS

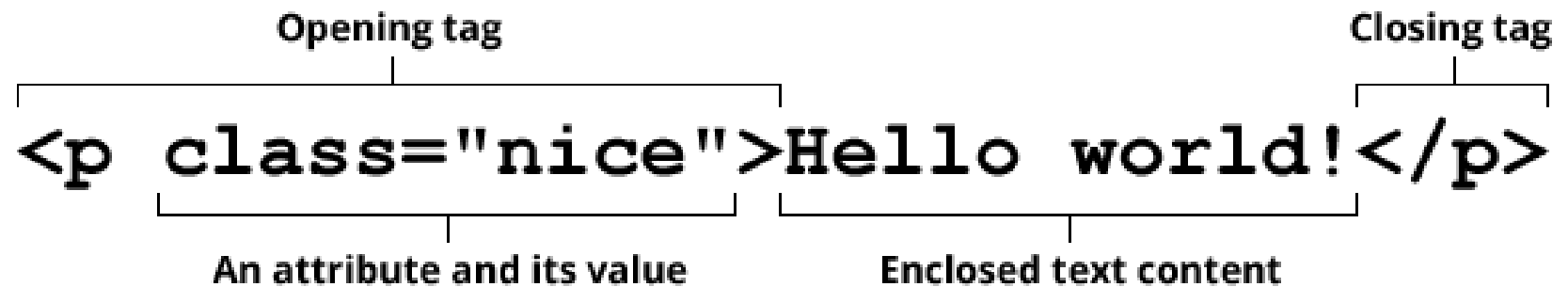


5

WHAT IS HTML?

- HTML is a markup language
- HTML separates "content" (words, images, audio, video, and so on) from "presentation" (instructions for displaying each type of content)
- HTML uses a pre-defined set of elements to define content types

Anatomy of an HTML element



ELEMENTS THE BASIC BUILDING BLOCKS

ATTRIBUTES

- An attribute extends a tag, changing tag behavior or providing metadata. An attribute always has the form `name=value` (giving the attribute's identifier and the attribute's associated value).

ATTRIBUTES

Named character references

- `>`; denotes the greater-than sign (`>`)
- `<`; denotes the less-than sign (`<`)
- `&`; denotes the ampersand (`&`)
- `"`; denotes double quote (`"`)

Comments and doctype

- `<!-- This is comment text -->`
- `<!DOCTYPE html>`

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <title>A tiny document</title>
5 </head>
6 <body>
7   <h1>Main heading in my document</h1>
8   <!-- Note that it is "h" + "1", not "h" + the letters "one" -->
9   <p>Look Ma, I am coding <abbr title="Hyper Text Markup Language">HTML</abbr>.</p>
10 </body>
11 </html>
```

A COMPLETE BUT SMALL DOCUMENT

```
1 | p {  
2 |   font-family: "Times New Roman", georgia, sans-serif;  
3 |   font-size: 24px;  
4 | }
```

CASCADING STYLE SHEET - CSS

CSS syntax consists of easy-to-use, intuitive keywords:

CASCADING THE RULES

- To cascade is to flow or follow downwards, in CSS this means that:
 - rules that are obeyed or discovered further down the chain override anything further up the chain

| % | percentage |
|----------|---|
| in | inch |
| cm | centimeter |
| mm | millimeter |
| em | 1em is equal to the current font size. 2em means 2 times the size of the current font. E.g., if an element is displayed with a font of 12 pt, then '2em' is 24 pt. The 'em' is a very useful unit in CSS, since it can adapt automatically to the font that the reader uses |
| rem | root em, equal to the em size of the root element (html) |
| ex | one ex is the x-height of a font (x-height is usually about half the font-size) |
| pt | point (1 pt is the same as 1/72 inch) |
| pc | pica (1 pc is the same as 12 points) |
| px | pixels (a dot on the computer screen) |

CSS UNITS



SELECTORS

- element selector
- group selector
- descendant selector
- child selector
- adjacent sibling
- general sibling

```
p { color: navy; }
```

```
p, ul, td, th { color: navy; }
```

```
li em { color: olive; }
```

```
h1 em, h2 em, h3 em { color: red; }
```

```
p > em {font-weight: bold;}
```

```
h1 + p {font-style: italic;}
```

```
h1 ~ h2 {font-weight: normal;}
```



SELECTORS

- id selector

```
li#catalog1234 { color: red; }  
#catalog1234 { color: red; }  
#links li { margin-left: 10px; }
```

- class selector

```
.special { color: orange; }  
p.special { color: orange; }
```

- universal selector

```
* {color: gray; }  
#intro * { color: gray; }
```



SPECIFICITY

- ID selectors are more specific than (and will override)
- Class selectors, which are more specific than (and will override)
- Contextual selectors, which are more specific than (and will override)
- Individual element selectors

```
strong { color: red;}  
h1 strong { color: blue; }
```

```
p { line-height: 1.2em; }  
blockquote p { line-height: 1em; }  
p.intro { line-height: 2em; }
```



MORE SELECTORS - PSEUDO-CLASS SELECTORS

■ Links

- :link
- :visited
- :hover
- :active

```
a:link {  
    color: maroon;  
}  
a:visited {  
    color: gray;  
}
```

■ Input

- :focus
- :checked
- :enabled
- :disabled

■ Text

- :first-line
- :first-letter

Snow White was ban
ate a poison apple, and
kissed her, married her,

■ Generate Content






- :before

```
p:before {  
    content: "Once upon a time: ";  
    font-weight: bold;  
    color: purple;  
}
```
- :after

Once upon a time: Snow White was banished f
seven dwarves, ate a poison apple, and fell aslee



ATTRIBUTE SELECTORS

- `element[title] {border: 3px solid;}`
- `element[title="first grade"] {border: 3px solid;}`
- `element[title~="grade"] {border: 3px solid;}`
- `element[attribute^="first part of the value"]
[src^="/images/icons"] {border: 3px solid;}`
- `element[attribute$="last part of the value"]
[href$=".pdf"] {border: 3px solid;}`



CSS SELECTORS

- http://www.w3schools.com/cssref/css_selectors.asp

<https://css-tricks.com/examples/ShapesOfCSS/>

KNOCKOUT COMPONENTS

- Components are a powerful, clean way of organizing your UI code into self-contained, reusable chunks.

```
ko.components.register('like-widget', {
  viewModel: function(params) {
    // Data: value is either null, 'like', or 'dislike'
    this.chosenValue = params.value;

    // Behaviors
    this.like = function() { this.chosenValue('like'); }.bind(this);
    this.dislike = function() { this.chosenValue('dislike'); }.bind(this);
  },
  template:
    '<div class="like-or-dislike" data-bind="visible: !chosenValue()">\
      <button data-bind="click: like">Like it</button>\
      <button data-bind="click: dislike">Dislike it</button>\
    </div>\
    <div class="result" data-bind="visible: chosenValue">\
      You <strong data-bind="text: chosenValue"></strong> it\
    </div>'
});
```

KNOCKOUT COMPONENTS

```
<ul data-bind="foreach: products">
  <li class="product">
    <strong data-bind="text: name"></strong>
    <like-widget params="value: userRating"></like-widget>
  </li>
</ul>
```

```
// Behaviors
this.like = function() { this.chosenValue('like'); }.bind(this);
this.dislike = function() { this.chosenValue('dislike'); }.bind(this);
},
template:
  '<div class="like-or-dislike" data-bind="visible: !chosenValue()">\
    <button data-bind="click: like">Like it</button>\
    <button data-bind="click: dislike">Dislike it</button>\
  </div>\
  <div class="result" data-bind="visible: chosenValue">\
    You <strong data-bind="text: chosenValue"></strong> it\
  </div>'
});
```

KNOCKOUT COMPONENTS AMD

```
ko.components.register('like-or-dislike', {  
  viewModel: { require: 'files/component-like-widget' },  
  template: { require: 'text!files/component-like-widget.html' }  
});
```

KNOCKOUT COMPONENTS AMD

```
ko.components.register('like-or-dislike', {  
  viewModel: { require: 'files/component-like-widget' },  
  template: { require: 'text!files/component-like-widget.html' }  
});
```

```
define(['knockout'], function(ko) {  
  return function(params) {  
    var chosenValue = params.value;  
    var like = function() {  
      chosenValue('like');  
    };  
    var dislike = function() {  
      chosenValue('dislike');  
    };  
  
    return {  
      chosenValue,      <div class="like-or-dislike" data-bind="visible: !chosenValue()">  
      like,              <button data-bind="click: like">Like it</button>  
      dislike,           <button data-bind="click: dislike">Dislike it</button>  
    };                  </div>  
  };  
});  
  
    <div class="result" data-bind="visible: chosenValue">  
      You <strong data-bind="text: chosenValue"></strong> it.  
      And this was loaded from an external file.  
    </div>
```

KNOCKOUT COMPONENTS AMD

```
ko.components.register('like-or-dislike', {  
  viewModel: { require: 'files/component-like-widget' },  
  template: { require: 'text!files/component-like-widget.html' }  
});
```

```
define(['knockout'], function(ko) {  
  return function(params) {  
    var chosenValue = params.value;  
    var like = function() {  
      chosenValue('like');  
    };  
    var dislike = function() {  
      chosenValue('dislike');  
    };  
  
    return {  
      chosenValue,  
      like,  
      dislike  
    };  
  };  
});
```

The require function

```
<div class="like-or-dislike" data-bind="visible: !chosenValue()">  
  <button data-bind="click: like">Like it</button>  
  <button data-bind="click: dislike">Dislike it</button>  
</div>  
  
<div class="result" data-bind="visible: chosenValue">  
  You <strong data-bind="text: chosenValue"></strong> it.  
  And this was loaded from an external file.  
</div>
```


KNOCKOUT COMPONENTS AMD

```
ko.components.register('like-or-dislike', {  
  viewModel: { require: 'files/component-like-widget' },  
  template: { require: 'text!files/component-like-widget.html' }  
});
```

```
define(['knockout'], function(ko) {  
  return function(params) {  
    var chosenValue = params.value;  
    var like = function() {  
      chosenValue('like');  
    };  
    var dislike = function() {  
      chosenValue('dislike');  
    };  
  
    return {  
      chosenValue,  
      like,  
      dislike  
    };  
  };  
});
```

The require function

Return the view model as a function

```
<div class="like-or-dislike" data-bind="visible: !chosenValue()">  
  <button data-bind="click: like">Like it</button>  
  <button data-bind="click: dislike">Dislike it</button>  
</div>  
  
<div class="result" data-bind="visible: chosenValue">  
  You <strong data-bind="text: chosenValue"></strong> it.  
  And this was loaded from an external file.  
</div>
```

WORD CLOUD



JQCLOUD