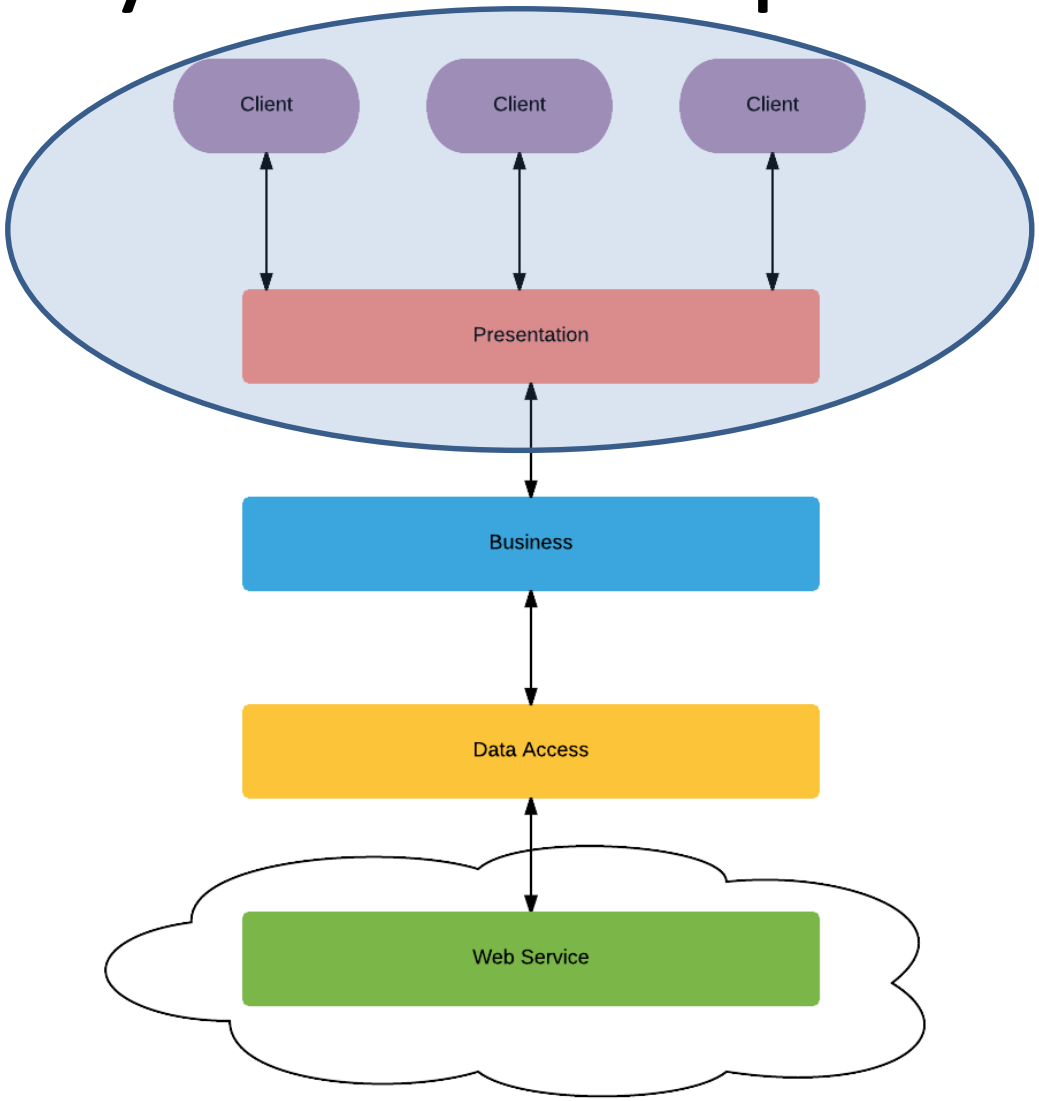
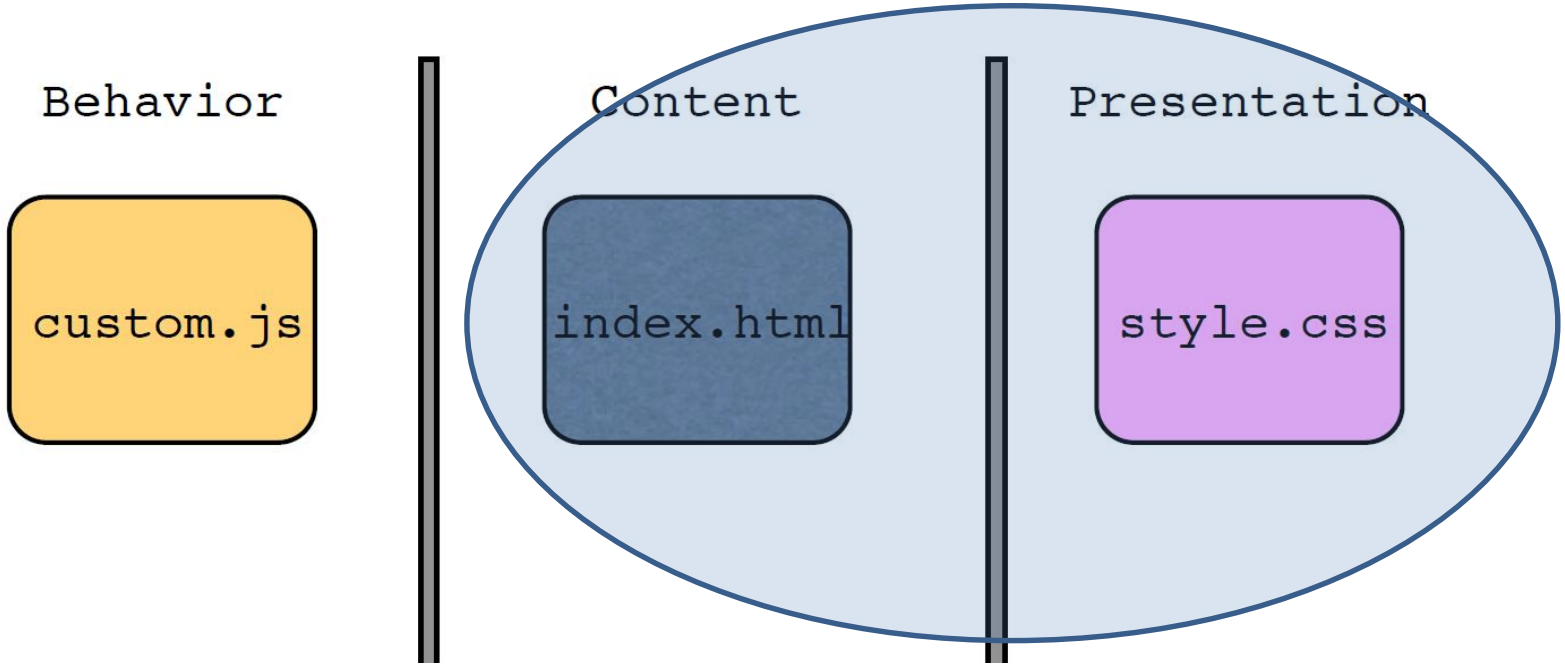
RAWDATA
Section 4

Troels Andreasen & Henrik Bulskov

System Development



Unobtrusive Design



Knockout

- Bindings
- Observables
- Templates and Control of Flow

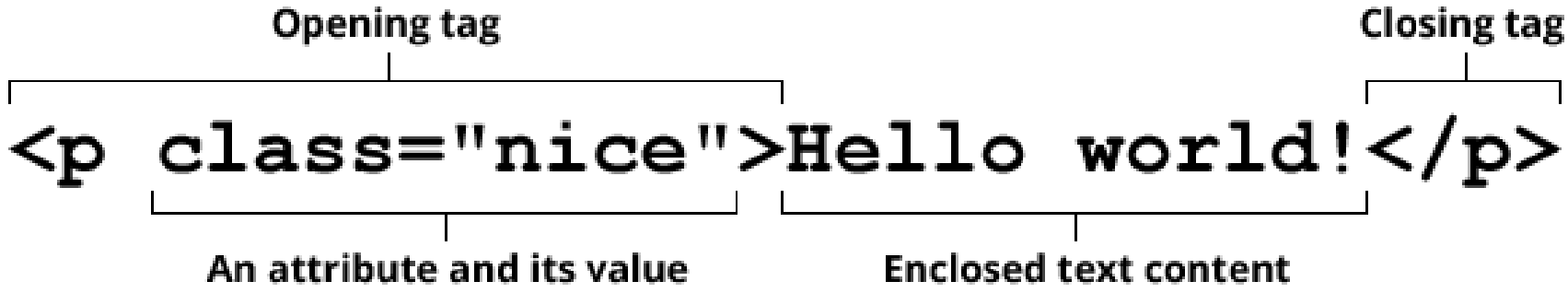
Key Knockout Concepts

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

Elements — the basic building blocks

Anatomy of an HTML element



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).

- 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>

A complete but small document

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

Cascading style sheet - CSS

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

```css
p {
  font-family: "Times New Roman", georgia, sans-serif;
  font-size: 24px;
}
```

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

CSS Units

%	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)

Selectors

- element selector
- group selector

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

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

- decendant selector

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

- child selector
- adjecent sibling
- generel sibling

```
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; }
```

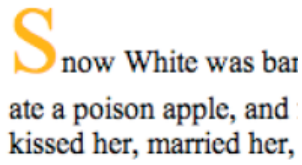
```
p { line-height: 1.2em; }
blockquote p { line-height: 1em; }
p.intro { line-height: 2em; }
```

More Selectors - Pseudo-Class Selectors

- Links
  - :link
  - :visited
  - :hover
  - :active
- Input
  - :focus
  - :checked
  - :enabled
  - :disabled

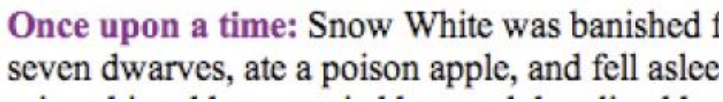
```
a:link {
  color: maroon;
}
a:visited {
  color: gray;
}
```

- Text
  - :first-line
  - :first-letter



- Generate Content
  - :before
  - :after

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



Attribute Selectors

- element[attribute]

```
img[title] {border: 3px solid;}
```

- element[attribute="exact value"]

```
img[title="first grade"] {border: 3px solid;}
```

- element[attribute~="value"]

```
img[title~="grade"] {border: 3px solid;}
```

- element[attribute^="first part of the value"]

```
img[src^="/images/icons"] {border: 3px solid;}
```

- element[attribute$="last part of the value"]

```
a[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>
```

```
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 AMD

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

http://knockoutjs.com/documentation/component-overview.html

```
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
        };
    }
});
```

```
<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>
```

```
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>
```

```
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


Nam et
et netus
et malesuada
diam
metus
dui
habitant
ut ultrices
Sit
Leo
sit amet
Sapien
adipiscing
vestibulum
Ipsum
Lorem
Dolor
elit
Aenean
Ut et leo
fames
metus
justo
Elit
tristisque
Adipiscing
Amet
Consectetur
morbi
egestas
purus
ac turpis
senectus
Pellentesque
et nulla
ullamcorper
dictum
at molestie

jQCloud