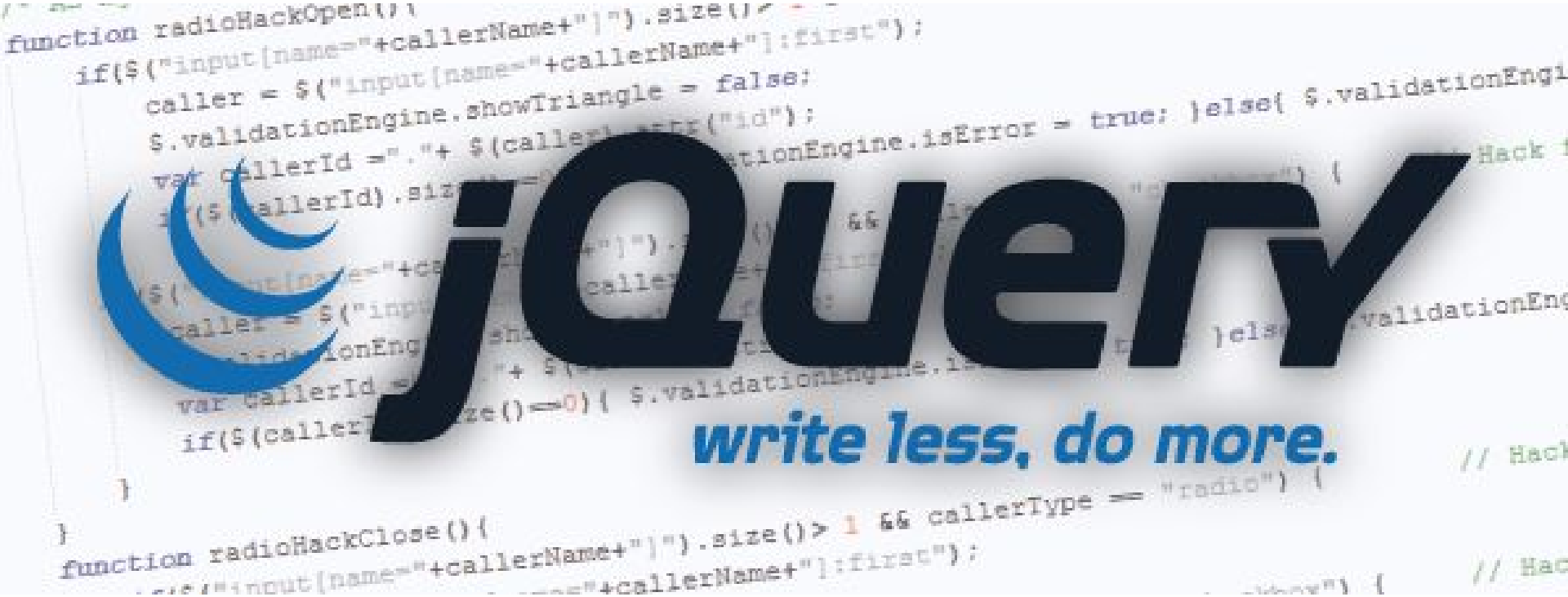
# Intro til jQuery




Hurtig demo af hvad det kan bruges til

# Tooling

Chrome DevTools er din BFF.

And together you're gonna run around, you're gonna... do all kinds of wonderful things. Just you and Chrome. The outside world is your enemy, you're the only.... friends you've got! It's just you and Chrome. You and Chrome and your adventures.. YOU AND CHROME FOREVER AND FOREVER A HUNDRED YEARS you and Chrome.. some...things.. You and yourself and Chrome runnin' around and... debugging... a- all day long forever.. all a - a hundred days debuggy-da-dup! forever a hundred times.... OVER and over... adventures dot com.. W W W dot at Chrome and DevTools dot com w..w..w... Chrome DevTools adventures.. ah- hundred years..... every minute DevTools dot com.... w w w a hundred times... JS BS dot com.....

Chrome extension - https://developer.chrome.com/devtools
Tips and Tricks - https://www.keycdn.com/blog/chrome-devtools/

# Hvorfor jQuery?

Webudvikling er svært nok i forvejen. Så vi bruger jQuery til at gøre det nemmere.

JQuery er en samling funktioner som gør det nemt at DOM manipulere, HTML document traversal, event håndtering, animationer og Ajax.

Men den bedste feature er at den har et cross-browser API. Dvs at hvis det virker i Chrome så virker det også i Internet Expl(💩)rer. *I teorien…*

Browserne er dog blevet meget bedre til at overholde standarderne så den pointe bliver mindre og mindre vigtig.

Men så er det godt at jQuery forsimpler API'erne så det er stadig smart :)

# jQuery på dit site

Man inkluderer jQuery via et <script> tag

Enten via lokal reference:
<script src="path/to/jquery.js"></script>

Eller via CDN.

```
<script
  src="https://code.jquery.com/jquery-3.2.1.min.js"
  integrity="sha256-hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwlAQgxVSNgt4="
  crossorigin="anonymous"></script>
```

Fordelen ved en CDN er at browsers cacher indholdet og når folk besøger din side så kan de have cached "https://code.jquery.com/jquery-3.2.1.min.js" og skal derfor ikke download det igen. Ergo din side loader hurtigere.

Ulempen er at en CDN kan gå ned og så virker dit site måske ikke.

Når det er gjort kan du bruge jQuery eller $ (alias for jQuery) på dit site.

# Hvor inkluderer man scripts?

```
<html>
<head>
   <script src="my-script.js"></script>
</head>
<body>
   <p>Hello, world!</p>
   <script src="my-script.js"></script>
</body>
</html>
```

Externe scripts blocker rendering mens det hentes og eksekveres.
Så placeret i <head> vil gøre at siden tager lang tid før den begynder at render.

I bunden (før </body></html>) vil forårsage at siden bliver vist hurtigt, men side kan "fryse" mens scripts loads og hvis de ændre på indholdet vil brugeren se et glimt af den ikke færdige side.

Løsningen er at bruge async og/eller defer, disse blocker ikke rendering. Virker dog kun i nyere browsere.
Async henter scripts asynkront og eksekvere det så snart det er hentet.
Defer henter og eksekvere scripts i rækkefølge.

Så den anbefalede måde er at loade scripts i <head> og bruge async/defer, for så virker det stadig i gamle browsere og de nye browsere vil få et hastigheds boost.

# jQuery ready()

Eftersom scripts eksekveres når de loades, som kan være før DOM'en er klar, bliver vi nødt til at sikre os at scriptet bliver kørt på det rigtige tidspunkt. Altså når DOM'en er klar. Heldigvis har jQuery løst det problem for os.

```
$(document).ready(function() {
  console.log( "ready!" );
});
```

Ikke at forveksle med window.onload = function() { ... }; som bliver kaldt når alt er loaded (billeder/iframes/osv).

```
// Shorthand for $( document ).ready()
$(function() {
  console.log( "ready!" );
});
```

Simpel regel. Pak alt jeres javascript ind i $(function() {...});

# jQuery selectors

**Select by ID.** Prefix med #, vil søge efter et element med ID'et. Husk at ID'er skal være unikke pr document.
$('#myId') -> [first or none]

**Select by class.** Prefix med punktum, vil søge efter elementer med den givne css class.
$('.myClass') -> [liste med elementer]

**Select by Attribute.** [name(~$!*)="value"]
$('[attr="value"]') -> [liste med elementer hvor værdien er præcis value]

**Multiple Selector**
$('selector1, selector2, selectorN') -> [liste med elementer som matcher en selector1-N]

**Descendant Selector**
$('ancestor descendant') -> [liste med descendants som har en ancestor som parent et sted i DOM træet]

# jQuery selectors (fortsat)

**:nth-child() Selector**
$('tr:nth-child(odd)') -> [liste med hver anden table row, rows er 1 indexed]

**Child Selector**
$('parent > child') -> [liste med children til parent]

**Next Adjacent Selector**
$('prev + next') -> [liste med next elementer som kommer lige efter prev]

**Next Siblings Selector**
$('prev ~ siblings') -> [liste med siblings som kommer efter prev og har samme parent]

Mange flere… http://api.jquery.com/category/selectors/

# jQuery DOM manipulation

**DOM iteration**

```
$(...).each(function(index) {
  // do all the things…
  $(this); // element ref
});
```

**DOM manipulation**

$(...).html() -> returnere html'en for det "selected" element.
$(...).html( htmlString ) -> sætter htmlString på det "selected" element.

$(...).append( content [, content ] ) -> tilføjer content til slutning af det "selected" element.

$(...).addClass( className ) -> tilføjer en css klasse til det "selected" element.

Mange flere… http://api.jquery.com/category/manipulation/

# jQuery events

Bind en handler (funktion) til HTML DOM Events
$(...).on( events [, selector ] [, data ], handler )

Der findes også forkortede versioner til de mest brugte events, som fx.
$(...).click( handler )

$(...).change( handler )

$(...).focus( handler )

$(...).blur( handler )

$(...).hover( handlerIn, handlerOut )

Og mange flere… http://api.jquery.com/category/events/

# jQuery Ajax (Asynchronous JavaScript And XML)

Bruges til at lave GET, POST osv. kald. Basis funktionen ser således ud:
$.ajax( url [, settings ] )

Der findes igen forkortede versioner til de mest brugte, som fx.

$.get( url [, data ] [, success ] [, dataType ] )
$.post( url [, data ] [, success ] [, dataType ] )

Disse returnere et Promise så man attacher en eller flere success/error handlers.

```
$.get(...).done(function(data) {
   // do stuff
}).fail(function(error) {
   // handle failure
});
```

Læs mere her http://api.jquery.com/category/ajax/