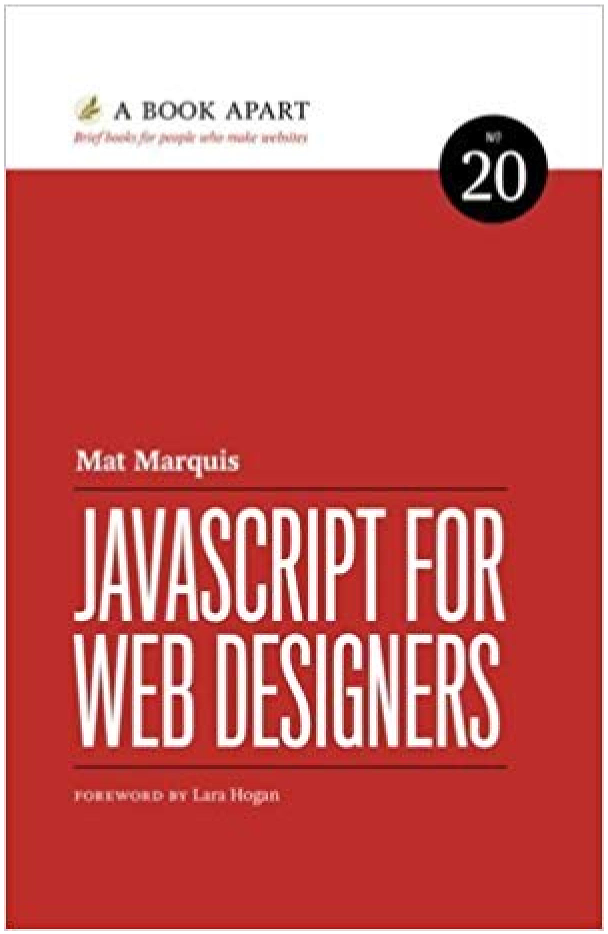
Introduction to JavaScript

Heinz Wittenbrink

2019-12-14

Intro

Caveat


A BOOK APART
Brief books for people who make websites
Nº 20
Mat Marquis
JAVASCRIPT FOR WEB DESIGNERS
FOREWORD BY Lara Hogan

Work in progress! In the moment it is mostly an overview of Mat Marquis' excellent Introduction to JavaSript for Designers

(Marquis, 2016)

We will make a lot of use of the code examples in the last chapter of Marquis' book.

Behaviour and dynamic content

JavaScript and the architecture of the web

- The web is an open and universal environment for digital content
- It is based on standards which guarantee the accessibilty and interoperability of contents
- To comply with these standards makes sense for ethical and political, but also for practical reasons

---

- HTML is the standard for textual content on the web
- CSS is the standard for the presentation of content on the web
- JavaScript (EcmaScript) is the standard for programmed manipulation of web content in the browser

Content, Presentation and Behaviour as Layers

> JavaScript allows us to add interaction to our pages as a complement to the structural layer that is markup and the presentational layer that is CSS.

(Marquis, 2016)

> The same kinds of web standards efforts that brought us semantically-meaningful markup and sane CSS support have also made JavaScript's syntax more consistent from browser to browser, and set reasonable constraints around the parts of a browser's behavior it can influence.

(Marquis, 2016)

Use cases for JavaSript

JavaScript for presentation

Lazy Loading Images and Video | Web Fundamentals | Google Developers

Lazy loading https://www.andreaverlicchi.eu/lazyload/

Dynamic loading of content

- Fetch API - Web APIs | MDN

- Using JavaScript Fetch API to Get and Post Data (Example)
- AJAX - Web-Entwickler Leitfäden | MDN

JavaScript for browser enhancement

- Modernizr: the feature detection library for HTML5/CSS3
- [Polyfill - MDN Web Docs Glossary: Definitions of Web-related terms | MDN](https://developer.mozilla.org/en-US/docs/Glossary/Polyfill "Polyfill - MDN Web Docs Glossary: Definitions of Web-related terms | MDN

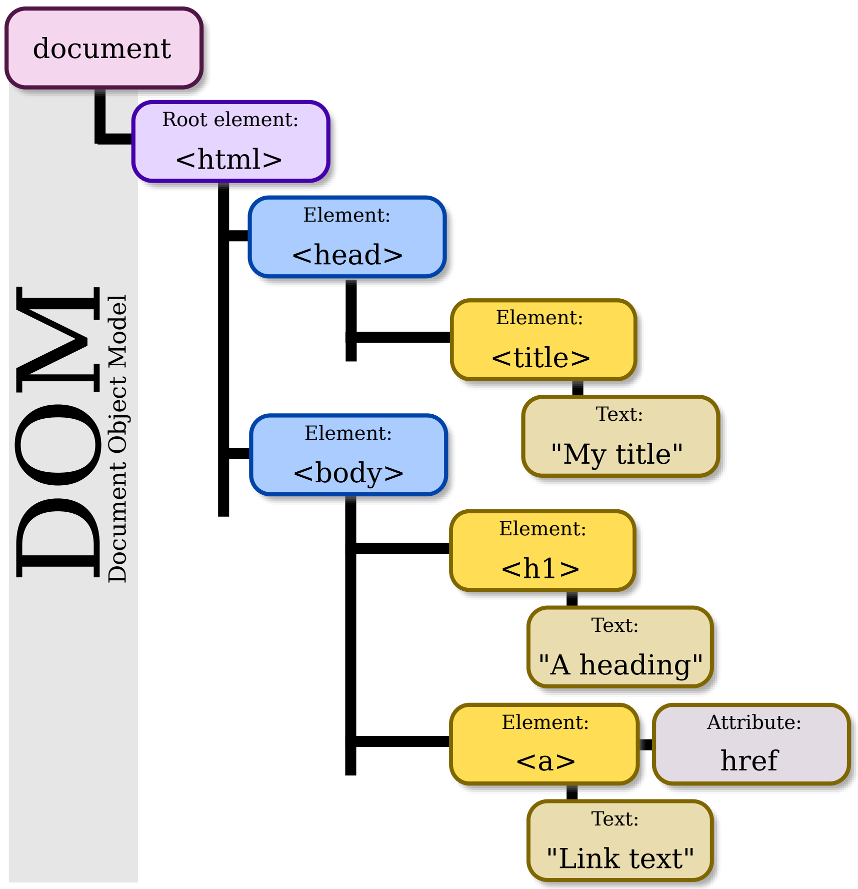
Extending HTML

- Web Components | MDN
- Custom Elements
- Custom Elements: defining new elements in HTML - HTML5 Rocks
- Polymer Project

Progressive Web Apps

Progressive Web Apps

DOM and DOM scripting



Birger Eriksson, https://commons.wikimedia.org/wiki/File:DOM-model.svg

> JavaScript communicates with the contents of our pages by way of an API named the Document Object Model, or DOM

(Marquis, 2016)

> the DOM API allows us to read, alter, and remove information from documents—to change things on the webpage itself.

(Marquis, 2016)

> The DOM serves two purposes. The first is providing JavaScript with a structured "map" of the page by translating our markup to a form that JavaScript (and many other languages) can understand.

(Marquis, 2016)

> Every element, comment, and even snippet of text is a node.

(Marquis, 2016)

> The DOM’s second purpose is to provide JavaScript with a set of methods and functions that allow access to the mapped nodes

(Marquis, 2016)

The language

Connecting content and scripts

> When it comes to JavaScript, though, we need to put a little more thought into placement, no matter whether the scripts are external or part of the page itself.

(Marquis, 2016)

> Including too many scripts in the head can make our pages feel slow.

(Marquis, 2016)

console

- With all modern browsers JavaScript can be written in the console and interactively executed
- The console gives access to the DOM of the document loaded in the open tab

---

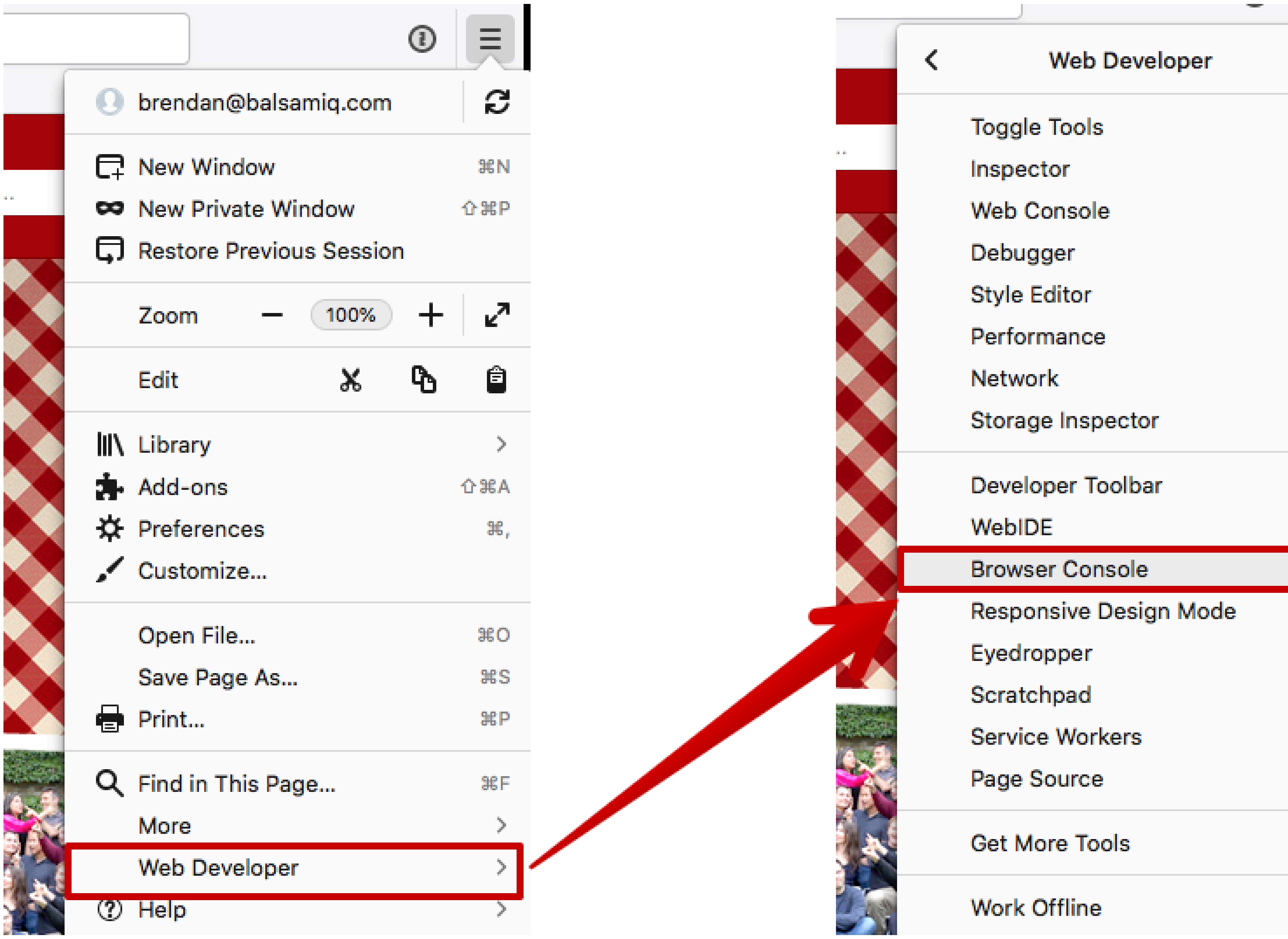
> These days, browsers compete on the quality of their dev tools, and we have tons of options for digging into the internals of our scripts—the simplest of which is still to send ourselves a message from inside the script, but with the potential to contain much more information than a simple string of text.

(Marquis, 2016)

Open Firefox Web Console:

press the `CtrlShiftK` (`Command` on OS X) keyboard shortcut.`OptionK`

[Open Chrome DevTools | Tools for Web Developers | Google Developers]



[Finding Your Browser's Developer Console | Balsamiq]

Example

[Sample page]

Creating a link

```
var newLink = document.createElement( 'a' );
```

Setting the attribute and its value

```
var newLink = document.createElement( 'a' );

newLink.setAttribute( 'href', '#' );

newLink.innerHTML = 'Read more';
```

Referencing the first paragraph

```
var newLink = document.createElement( 'a' );

var allParagraphs = document.getElementsByTagName( 'p' );

var firstParagraph = allParagraphs[ 0 ];

newLink.setAttribute( 'href', '#' );

newLink.innerHTML = 'Read more';
```

Appending the link as a child

```
var newLink = document.createElement( 'a' );

var allParagraphs = document.getElementsByTagName( 'p' );

var firstParagraph = allParagraphs[ 0 ];

newLink.setAttribute( 'href', '#' );

newLink.innerHTML = 'Read more';

firstParagraph.appendChild( newLink );
```

Changing the display of the link

```
var newLink = document.createElement( 'a' );

var allParagraphs = document.getElementsByTagName( 'p' );
var firstParagraph = allParagraphs[ 0 ];

newLink.setAttribute( 'href', '#' );

newLink.innerHTML = 'Read more';

newLink.style.display = 'inline-block';

newLink.style.marginLeft = '10px';

firstParagraph.appendChild( newLink );
```

Delegating the display to a stylesheet

```
var newLink = document.createElement( 'a' );

var allParagraphs = document.getElementsByTagName( 'p' );

var firstParagraph = allParagraphs[ 0 ];

newLink.setAttribute( 'href', '#' );

newLink.setAttribute( 'class', 'more-link' );

newLink.innerHTML = 'Read more';

firstParagraph.appendChild( newLink );
```

Logging all paragraphs

```
var newLink = document.createElement( 'a' );

var allParagraphs = document.getElementsByTagName( 'p' );

var firstParagraph = allParagraphs[ 0 ];

newLink.setAttribute( 'href', '#' );

newLink.setAttribute( 'class', 'more-link' );

newLink.innerHTML = 'Read more';
```

```
for( var i = 0; i < allParagraphs.length; i++ ) {

  console.log( allParagraphs[ i ] );

}

firstParagraph.appendChild( newLink );
```

Hiding the other paragraphs

```
var newLink = document.createElement( 'a' );

var allParagraphs = document.getElementsByTagName( 'p' );

var firstParagraph = allParagraphs[ 0 ];

newLink.setAttribute( 'href', '#' );

newLink.setAttribute( 'class', 'more-link' );

newLink.innerHTML = 'Read more';

for( var i = 0; i < allParagraphs.length; i++ ) {

  allParagraphs[ i ].style.display = 'none';

}

firstParagraph.appendChild( newLink );
```

… but showing the first paragraph

```
var newLink = document.createElement( 'a' );

var allParagraphs = document.getElementsByTagName( 'p' );

var firstParagraph = allParagraphs[ 0 ];

newLink.setAttribute( 'href', '#' );
newLink.setAttribute( 'class', 'more-link' );

newLink.innerHTML = 'Read more';

for( var i = 0; i < allParagraphs.length; i++ ) {

  if( i === 0 ) {

    continue;

  }

  allParagraphs[ i ].style.display = 'none';

}

firstParagraph.appendChild( newLink );
```

Adding an event listener and a function

```
var newLink = document.createElement( 'a' );

var allParagraphs = document.getElementsByTagName( 'p' );

var firstParagraph = allParagraphs[ 0 ];

function revealCopy() {

  console.log( 'Clicked!' );

}

newLink.setAttribute( 'href', '#' );

newLink.setAttribute( 'class', 'more-link' );

newLink.innerHTML = 'Read more';

newLink.addEventListener( 'click', revealCopy );

for( var i = 0; i < allParagraphs.length; i++ ) {
  if( i === 0 ) {
    continue;
  }
  allParagraphs[ i ].style.display = 'none';
}
firstParagraph.appendChild( newLink );
```

Adding the click event as an argument

```
var newLink = document.createElement( 'a' );
var allParagraphs = document.getElementsByTagName( 'p' );
var firstParagraph = allParagraphs[ 0 ];
function revealCopy( e ) {
  console.log( e );
}
newLink.setAttribute( 'href', '#' );
newLink.setAttribute( 'class', 'more-link' );
newLink.innerHTML = 'Read more';
newLink.addEventListener( 'click', revealCopy );
for( var i = 0; i < allParagraphs.length; i++ ) {
  if( i === 0 ) {
    continue;
  }
  allParagraphs[ i ].style.display = 'none';

}

firstParagraph.appendChild( newLink );
```

Preventing the link’s default behaviour

```
var newLink = document.createElement( 'a' );

var allParagraphs = document.getElementsByTagName( 'p' );

var firstParagraph = allParagraphs[ 0 ];

function revealCopy( e ) {

  e.preventDefault();

};

newLink.setAttribute( 'href', '#' );

newLink.setAttribute( 'class', 'more-link' );

newLink.innerHTML = 'Read more';

newLink.addEventListener( 'click', revealCopy );

for( var i = 0; i < allParagraphs.length; i++ ) {

  if( i === 0 ) {

    continue;

  }

  allParagraphs[ i ].style.display = 'none';

}

firstParagraph.appendChild( newLink );
```

Changing the display of the hidden paragraphs

```
var newLink = document.createElement( 'a' );

var allParagraphs = document.getElementsByTagName( 'p' );

var firstParagraph = allParagraphs[ 0 ];

function revealCopy( e ) {

  var allParagraphs = document.getElementsByTagName(   'p' );

  for( var i = 0; i < allParagraphs.length; i++ ) {

    if( i === 0 ) {

      continue;
    }

    allParagraphs[ i ].style.display = 'block';

   }

   e.preventDefault();

}

newLink.setAttribute( 'href', '#' );

newLink.setAttribute( 'class', 'more-link' );

newLink.innerHTML = 'Read more';

newLink.addEventListener( 'click', revealCopy );

for( var i = 0; i < allParagraphs.length; i++ ) {

if( i === 0 ) {

continue;

}
allParagraphs[ i ].style.display = 'none';

}

firstParagraph.appendChild( newLink );
```

Removing the link

```
var newLink = document.createElement( 'a' );

var allParagraphs = document.getElementsByTagName( 'p' );

var firstParagraph = allParagraphs[ 0 ];

function revealCopy( e ) {

  var allParagraphs = document.getElementsByTagName(   'p' );

  for( var i = 0; i < allParagraphs.length; i++ ) {

    if( i === 0 ) {

      continue;

    }

    allParagraphs[ i ].style.display = 'block';

  }

  this.remove();

  e.preventDefault();

}

newLink.setAttribute( 'href', '#' );

newLink.setAttribute( 'class', 'more-link' );

newLink.innerHTML = 'Read more';
newLink.addEventListener( 'click', revealCopy );

for( var i = 0; i < allParagraphs.length; i++ ) {

  if( i === 0 ) {

    continue;

  }

  allParagraphs[ i ].style.display = 'none';

}

firstParagraph.appendChild( newLink );
```

Optimizing 1: Renaming the function

```
var newLink = document.createElement( 'a' );

var allParagraphs = document.getElementsByTagName( 'p' );

var firstParagraph = allParagraphs[ 0 ];

function toggleCopy( e ) {

  var allParagraphs = document.getElementsByTagName(    'p' );

  for( var i = 0; i < allParagraphs.length; i++ ) {

    if( i === 0 ) {

      continue;

    }

    allParagraphs[ i ].style.display = 'block';

  }

  this.remove();
  e.preventDefault();
}
newLink.setAttribute( 'href', '#' );
newLink.setAttribute( 'class', 'more-link' );
newLink.innerHTML = 'Read more';
newLink.addEventListener( 'click', toggleCopy );
toggleCopy();
firstParagraph.appendChild( newLink );
```

Optimizing 2:Making sure that this is referencing the link

```
var newLink = document.createElement( 'a' );
var allParagraphs = document.getElementsByTagName( 'p' );
var firstParagraph = allParagraphs[ 0 ];
function toggleCopy( e ) {
  var allParagraphs = document.getElementsByTagName(   'p' );
  for( var i = 0; i < allParagraphs.length; i++ ) {
    if( i === 0 ) {
      continue;
    }
    allParagraphs[ i ].style.display = 'block';
    }
    if( this === newLink ) {

      this.remove();

    }

  e.preventDefault();

}

newLink.setAttribute( 'href', '#' );

newLink.setAttribute( 'class', 'more-link' );

newLink.innerHTML = 'Read more';

newLink.addEventListener( 'click', toggleCopy );

toggleCopy();

firstParagraph.appendChild( newLink );
```

Optimizing 3: Making sure that e refers to a defined event

```
var newLink = document.createElement( 'a' );

var allParagraphs = document.getElementsByTagName( 'p' );

var firstParagraph = allParagraphs[ 0 ];

function toggleCopy( e ) {

  var allParagraphs = document.getElementsByTagName(   'p' );

  for( var i = 0; i < allParagraphs.length; i++ ) {

    if( i === 0 ) {

      continue;
    }
    allParagraphs[ i ].style.display = 'block';
  }
  if( this === newLink ) {
    this.remove();
  }
  if( e !== undefined ) {
    e.preventDefault();
  }
}
newLink.setAttribute( 'href', '#' );
newLink.setAttribute( 'class', 'more-link' );
newLink.innerHTML = 'Read more';
newLink.addEventListener( 'click', toggleCopy );
toggleCopy();
firstParagraph.appendChild( newLink );
```

Optimizing 4: Making sure that e has a preventDefault() method

```
var newLink = document.createElement( 'a' );
var allParagraphs = document.getElementsByTagName( 'p' );
var firstParagraph = allParagraphs[ 0 ];
```

```
function toggleCopy( e ) {

  var allParagraphs = document.getElementsByTagName(   'p' );

  for( var i = 0; i < allParagraphs.length; i++ ) {

    if( i === 0 ) {

      continue;

    }

    allParagraphs[ i ].style.display = 'block';

  }

  if( this === newLink ) {

    this.remove();

  }

  if( e !== undefined && e.preventDefault !==   undefined ) {

    e.preventDefault();

  }

}

newLink.setAttribute( 'href', '#' );

newLink.setAttribute( 'class', 'more-link' );

newLink.innerHTML = 'Read more';

newLink.addEventListener( 'click', toggleCopy );

toggleCopy();

firstParagraph.appendChild( newLink );
```

Optimizing 5: Setting only the display of hidden blocks to block

```
var newLink = document.createElement( 'a' );

var allParagraphs = document.getElementsByTagName( 'p' );

var firstParagraph = allParagraphs[ 0 ];

function toggleCopy( e ) {

  var allParagraphs = document.getElementsByTagName(   'p' );

  for( var i = 0; i < allParagraphs.length; i++ ) {

    if( i === 0 ) {

      continue;

    }

    if( allParagraphs[ i ].style.display === 'none' ) {

      allParagraphs[ i ].style.display = 'block';

    } else {

      allParagraphs[ i ].style.display = 'none';

    }

  }

  if( this === newLink ) {

    this.remove();

  }

  if( e !== undefined && e.preventDefault !==   undefined ) {
    e.preventDefault();

  }

}

newLink.setAttribute( 'href', '#' );

newLink.setAttribute( 'class', 'more-link' );

newLink.innerHTML = 'Read more';

newLink.addEventListener( 'click', toggleCopy );

toggleCopy();

firstParagraph.appendChild( newLink );
```

Optimizing 6: Avoiding the repetition of the array for the paragraphs

```
var newLink = document.createElement( 'a' );

var allParagraphs = document.getElementsByTagName( 'p' );

var firstParagraph = allParagraphs[ 0 ];

function toggleCopy( e ) {

  var allParagraphs = document.getElementsByTagName(   'p' );

  for( var i = 0; i < allParagraphs.length; i++ ) {

    var para = allParagraphs[ i ];

    if( i === 0 ) {

      continue;

    }
    if( para.style.display === 'none' ) {

    para.style.display = 'block';

    } else {

    para.style.display = 'none';

    }

  }

  if( this === newLink ) {

    this.remove();

  }

  if( e !== undefined && e.preventDefault !==   undefined ) {

    e.preventDefault();

  }

}

newLink.setAttribute( 'href', '#' );

newLink.setAttribute( 'class', 'more-link' );

newLink.innerHTML = 'Read more';

newLink.addEventListener( 'click', toggleCopy );

toggleCopy();

firstParagraph.appendChild( newLink );
```

Final script: Not cluttering up the global scope

```
(function() {
```

```
var newLink = document.createElement( 'a' );

var allParagraphs = document.getElementsByTagName(   'p' );

var firstParagraph = allParagraphs[ 0 ];

function toggleCopy( e ) {

  var allParagraphs = document.getElementsByTagName(     'p' );

  for( var i = 0; i < allParagraphs.length; i++ ) {

    var para = allParagraphs[ i ];

    if( i === 0 ) {

      continue;

    }

    if( para.style.display === 'none' ) {

      para.style.display = 'block';

    } else {

      para.style.display = 'none';

    }

  }

  if( this === newLink ) {

    this.remove();

  }

  if( e !== undefined && e.preventDefault !==     undefined ) {

    e.preventDefault();
    }

  };

  newLink.setAttribute( 'href', '#' );

  newLink.setAttribute( 'class', 'more-link' );

  newLink.innerHTML = 'Read more';

  newLink.addEventListener( 'click', toggleCopy );

  toggleCopy();

  firstParagraph.appendChild( newLink );

}());
```

What is a scripting language?

- Interpreted, not compiled
- dynamic types

JSON

JSON Example

JavaScript frameworks

- Angular
- React – A JavaScript library for building user interfaces
- Vue.js

Progressive Enhancement

> Progressive enhancement is a strategy for web design that emphasizes core webpage content first. This strategy then progressively adds more nuanced and technically rigorous layers of presentation and features on top of the content as the end-user's browser/internet connection allow.

> The proposed benefits of this strategy are that it allows everyone to access the basic content and functionality of a web page, using any browser or Internet connection, while also providing an enhanced version of the page to those with more advanced browser software or greater bandwidth.

Serverside JavaScript

Node.js

JavaScript best practices - W3C Wiki https://www.w3.org/wiki/JavaScript_best_practices

https://www.andreaverlicchi.eu/lazy-load-responsive-images-in-2019-srcset-sizes-more/

Marquis, M. (2016). *Javascript for web designers*. New York, New York: A Book Apart.