IT2805 Web Technology Compendium

Adrian Opheim

December 13, 2017

Contents

1 Lecture 12.sept

Selecting the elements you want can be done like:

```
h1, h2, h3 {
...//Group selector
}

p#1234 {
... //Class selector
}

p a {
... // Selecting all a elements that are children of p. Targets any <a>
    elements that sit inside a <p> element, even if there are no
    elements nested between them.
}

one > two {
... // Selects an element that are the immediate child of one.
}
```

1.1 Pseudo-class

Selects elements based on the state they are in.

```
a:visited {
...// all the links that are visited by the user.
}

a:hover {
... // Objects that the user is hovering the mouse cursor over.
}
```

1.2 Pseudo-element

This represents elements in the HTML page

```
p::first-letter {
... //
}
```

1.3 Resolving a rule conflict

```
p {
color:blue !important
}
```

1.4 Overflowing content

2 Øvingsforelesning 18.sept

- `static` er standard-posisjoneringsverdi for HTML-elementer. Kan ses fra developer tools. Elementer kommer i rekkefølgen de er definert i.

- `position: relative;` gir mulighet til å spesifisere `top, bottom, left, right`. `top: 15px` vil flytte hele elementet 15 piksler ned fra dets origo. `relative` forstyrrer ingen andre elementer i dokumentflyten.
- `position: absolute` vil forstyrre dokumentflyten. Ser på første parent-element som er satt til relative. Forflytter seg etter det. Finner den ingen parent med relative, ser den på html-elementet (eller det øverste elementet). Kun hvis man bruker `absolute` kan man bruke Z-indeks for å bestemme hvillke elementer som ligger oppå hvilke.
- `position: fixed` kan brukes til å feste elementet til viewporten (alt vi ser i nettleservinduet).

3 Lecture 1: History, development and the architecture of the Web, Client, Server and the Internet

3.1 The Client

- Accesses the Web via software known as the browser, which locate and dosplay information from the web.
- Communication is done with an agreed protocol or transmission language, e.g. HTTP (Hyper-Text Tranfer Protocol)
- The browser communicates with the server, which delievers a web page, which is typically a text file containing HTML. This page is rendered in the browser.

3.2 The Server

- Responds to client requests for web pages. The web pages exist on its local file system. These pages are retrieved from the server, and then transmitted to the client.
- The **network** is a structure linking computers together for the purpose of sharing resources such as printers and files. A computer that makes a service availabe to a network is called a server. The users typically access a network through a computer called a **host** or **node**.
- LAN: Local Area Network: Computers that make up a network are close together.
- WAN: Network covering a wide area. The largest WAN in existence is the Internet. Internet started as ARPANET, developed for scientific and military use. From 1969.
- The physical structure of the internet uses fiber-optic cables, satellites, phone lines and other telecommuniacations media.
- The Internet is a network of interconnected networks. Meaning that if parts of its infrastructure is destroyed, data can still flow through the remaining networks.

3.3 Protocols

- Computers must speak a common language. The rules and procedures are defined in the protocols.

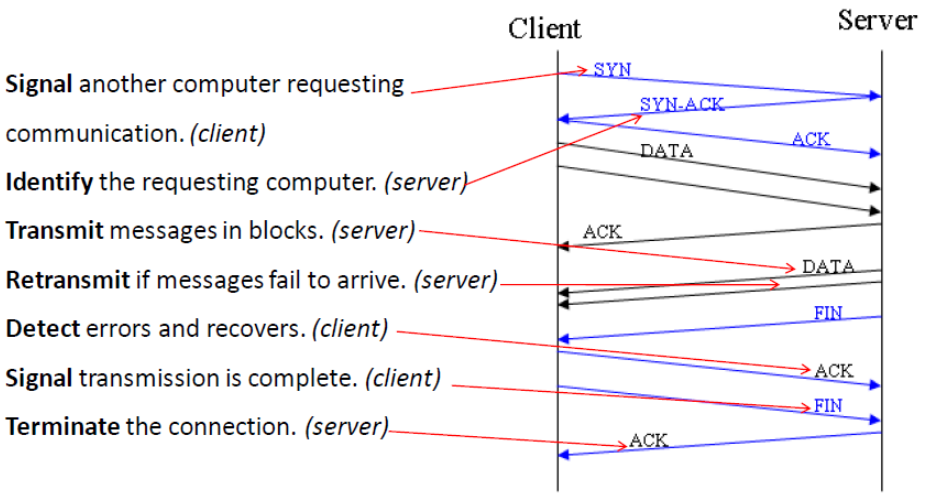

Figure 1: How communication is done, according to protocols.

- Every computer and network on the Internet uses the same protocols: the Transmission Control Protocol/Internet Protocol: TCP/IP. This means that as long as your computer system uses TCP/IP, it can exchange data with any other type of computer.
  TCP/IP layers:

  - **Application layer**: Provides applications the ability to access the services of the other layers and defines the protocols that applications use to exchange data.
    Ex.: HTTP, telnet, ftp, email, VoIP
  - **Transport Layer**: Makes sure that the complete message is delievered to the end.
    Ex.: TCP, UDP
  - **Network/Internet layer**: Routes messages from one place to another. All routers on the Internet run the IP protocol.
    Ex.: IP
  - **Physical layer**: Translating the software message into a physical representation and putting them on the wire.
    Ex.: Ethernet, WiFi, ATM, Frame Relay

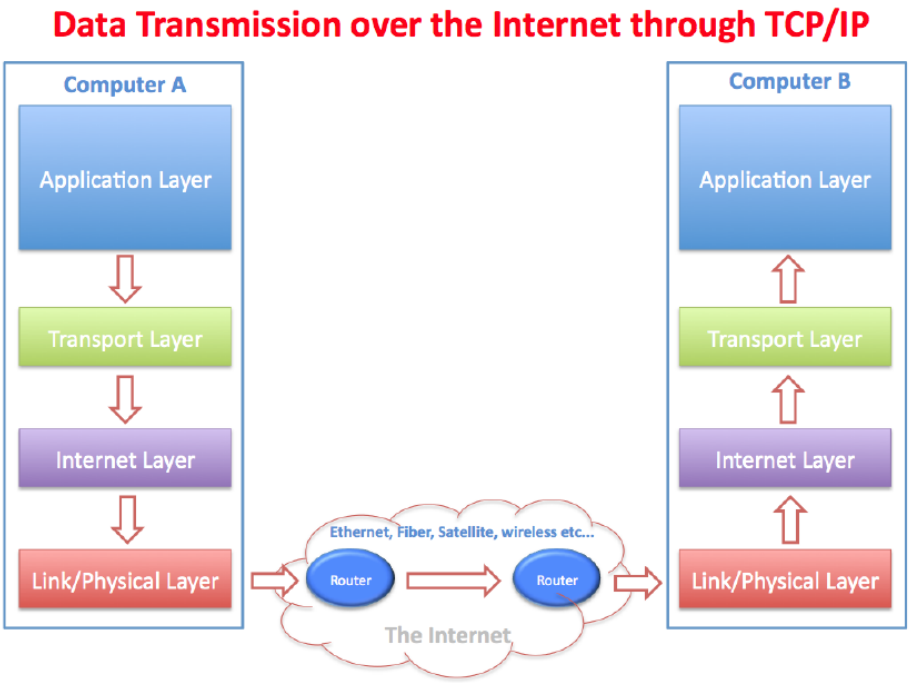

Figure 2: TCP/IP communication

- Every computer on the internet has a unique numeric identifier, called an Internet Protocol (IP) address. Ex.: 129.241.160.102. A domain is the word addess of the IP address.
- A Domain Name Server (DNS) computer work as a "phone list" that maps symbolic names to their IP.
- When using the Internet, requests and data are broken into packets and travel across multiple networks before being reassembled at their destination.
- **Dynamic routing**: The routes through the network is selected at the time of transmission, after considering current network conditions. This means that the network can not be hierarchical, as the failure of a parent will bring down the entire network.

3.4 Hypertext Documents

- Contain elements known as hyperlinks or links, that allow you to jump from one topic to another. The hypertext documents within a web site are known as web pages.

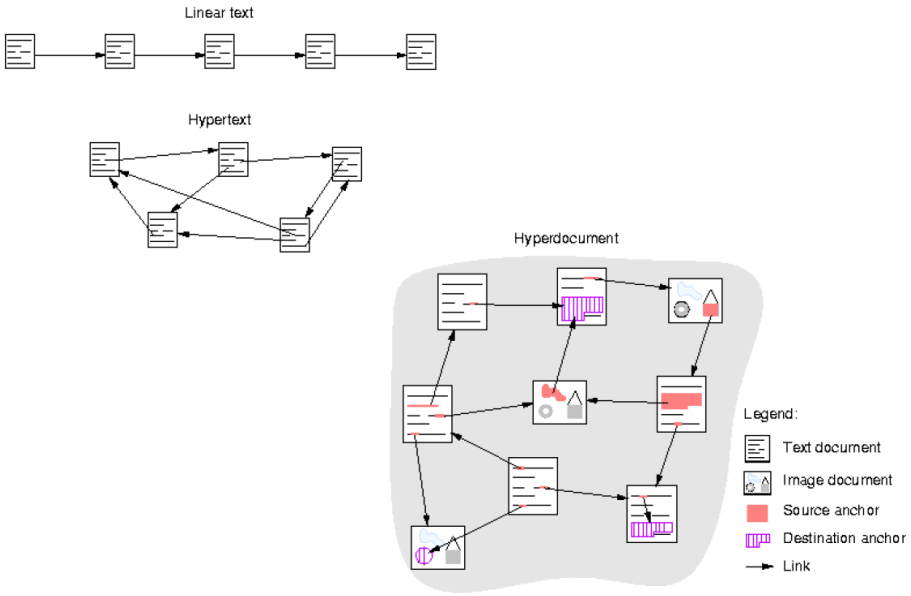

Figure 3: Hypertext documents

- The language for expressing hypertext is HyperText Markup Language (HTML).

3.5 HTML

- HTML is standardized by the World Wide Web Consortium (W3C). HTML5 is the newest HTML specification.

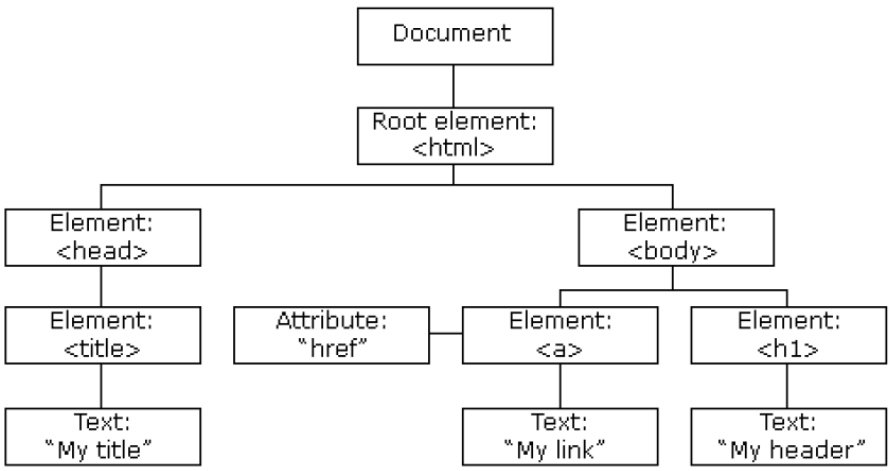

Figure 4: The HTML document tree

- A HTML document consists of a tree of HTML elements, where each element begins with a start tag, and end with a end tag.

```
<p>Hello world</p>
```

The mandatory elements in a HTML document: **Body, Head and Title**

- `<head>`: Contains information about the page. You will usually find a `<title>` element inside the `<head>` element.
- `<title>`:
- `<body>`: Everything inside this element is shown inside the main browser window.

Example:

```
<html>
    <head>
        <title>This is the Title of the page</title>
    </head>
    <body>
        <h1>This is a heading in the body of the page</h1>
        <p>Anything within the body of a web page is displayed in
            the main browser window.</p>
    </body>
</html>
```

- Web pages should be saved in plain text with `.htm` or `.html` filename extensions.

3.5.1 Attributes

- Indicates what kind of extra information you are supplying about the element's content. The value is the information or setting for the attribute. Should be placed in double quotes.
- The majority of attributes can only be used on certain elements, although a few attributes (such as `lang`) can appear on any element.

```
<p lang="en-us">Paragraph in English</p>
```

3.5.2 Doctype Declaration

- The `<!DOCTYPE>` declaration helps the the browser to display a web page correctly. For HTML5:

```
<!DOCTYPE html
```

For HTML4.01:

```
<!DOCTYPE HTML PUBLIC
```

4 Lecture 2: HTML, Basics, Tables and Page Design Issues

4.1 HTML file structure

- An HTML document is divided into two parts: the `head` and the `body`. **Example:**

```
<!DOCTYPE html> <!--Telling the browser what syntax to use-->
<html>
    <head>
        <meta charset="UTF-8"> <!-- Specifies character encoding,
            required in HTML5-->
        <title> My first HTML document </title>
    </head>

    <body>
        <p>Hello world!</p>
```

```
10      </body>
11  </html>
```

- Examples of metadata:

```
1  <head>
2      <meta name="description" content="lecture 3, examples">
3      <meta name="keywords" content="HTML, meta information">
4      <meta name="author" content="mike">
5      <meta charset="UTF-8">
6  </head>
```

- HTML tags can be two-sided or one-sided. General syntax for two-sided tag:

```
<element>Content</element>
```

- Element names could be uppercase, but it is good habit to use lowercase.
- A one-sided tag contains no content:

```
<element/>
```

- Elements that employ one-sided tags are called empty elements since they contain no content. Examples: Line-break: `<br/>`, horizontal line: `<hr/>`
- Commenting:

```
<!--This is a comment-->
```

- White-space are blank spaces, tabs and line breaks in the HTML file. These are treated as a single blank space, meaning that you can not format the HTML with blank spaces. Is overwritten with the `pre` tag:

```
1  <pre>This is preformatted text.
2  It preserves both spaces     and line breaks.</pre>
```

- A HTML element is either block or inline.
- **Inline** ex.: `<b>, <td>, <a>, <img>`
- **Block elements** start with a new line when displayed in a browser. Ex.: `<h1>, <p>, <ul>, <table>`

4.1.1 The `<span>` element

- An inline element that can be used as a container for text. Has no special meaning.
- When used with CSS, it can be used to set attributes to parts of text.

4.1.2 The `<div>` element

- A block-level element that can be used as a container for grouping other HTML elements.

- When used together with CSS, it can be used to set style attributes to large blocks of content.

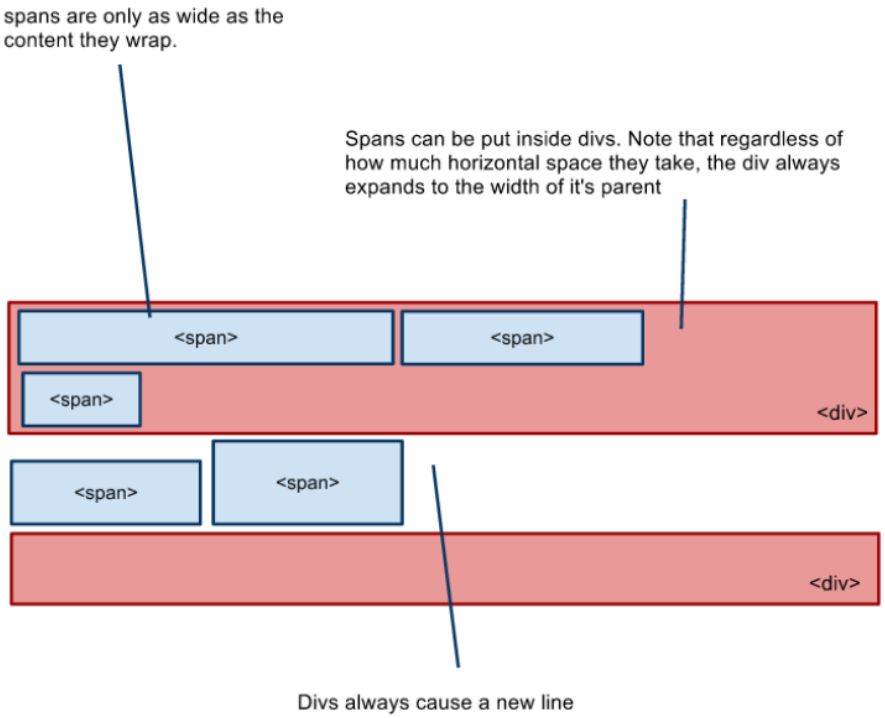

Figure 5: How span and div are used for grouping tags

4.1.3 Headings

- Divides the web page into parts. 6 levels of headings. Ex.: `<h1>This is a main heading</h1>`

4.1.4 Semantic Markup

- Semantic markup are for example `<strong>` or `<em>`, where `<em>` will show it content in italics.

4.1.5 Lists

- Three kinds of lists are supported:
  - `<ol>`: Ordered lists
  - `<ul>`: Unordered lists
  - `<dl>`: Definition lists for items that describe a term
- One list can contain another list, then being called a nested list.

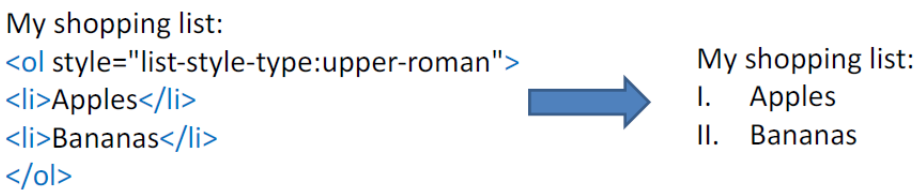

Figure 6: Using different styles for lists.

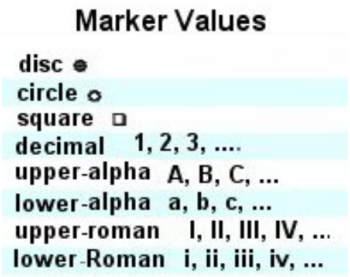

Figure 7: Possible marker values

- Defintion lists consist of a defining term: `<dt>` and a definition description: `<dd>`

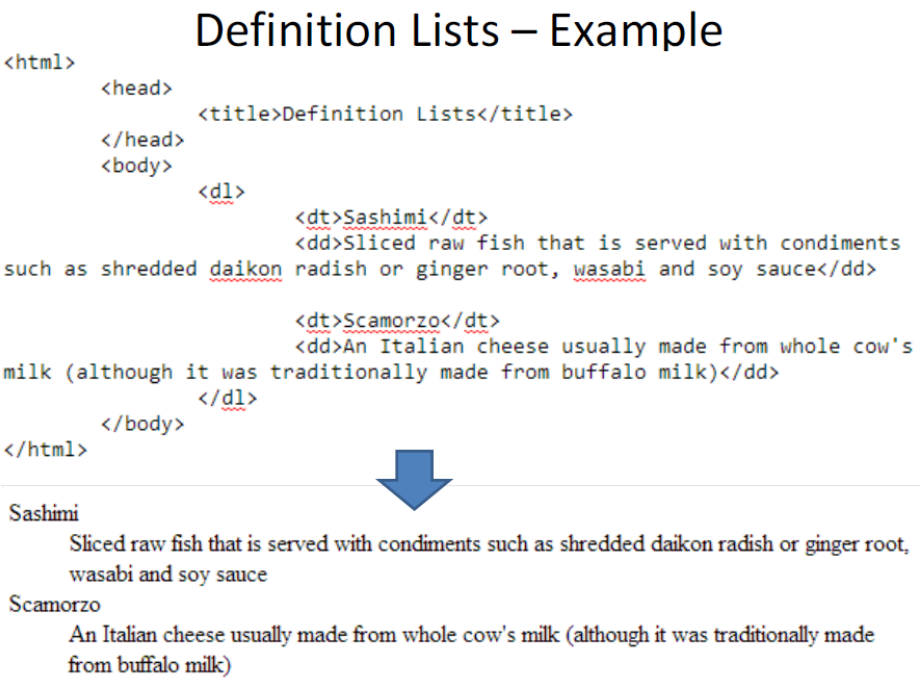

Figure 8: Example of definition lists usage

4.1.6 Links

- Links are created using the `<a>` element.

```
<a href="http://www.google.com">Link til Google</a>
```

- Opening links in a new window: `<a href="www.vg.no" target="_blank">VG</a>`
- By giving headers and `id`, you can link to specific parts of the same page:

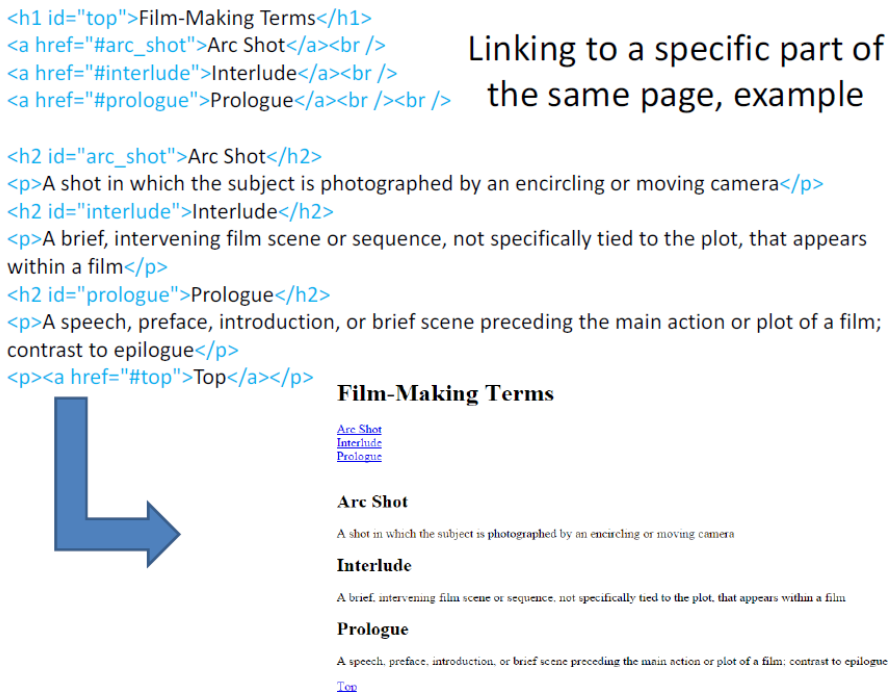

Figure 9: Linking internally in a HTML page

- This also works in URLs: If headings are marked with an id, they can be linked to in the URL, like: `www.sundance.org/#jumbotron`

4.1.7 Images

- Images are added by using `<img>`. The `img` element must contain:
  - `src`: tells the browser where it can find the image file
  - `alt`: this provides a text description of the image, for accessibility and indexing purposes.
  - `height` specifies height of the image in pixels
  - `width`: width of image in pixels. (Normally controlled with CSS)

```
<img src="images/dog.jpg" alt="dog (Pomeranian)"/>
```

  - Measure images in pixels (not centimeters or inches). Doing this, you will be independent from screen resolution.
  - Figure captions are introduced in HTML5 by using the `<figure>` element:

```
1 <figure>
2     <img src="images/image1.jpg" alt="the first image"/>
3     <figcaption>This is the caption of the first image</
          figcaption>
4 </figure>
```

4.1.8 Tables

```
<html><head>
        <title>Table Headings</title>
</head> <body>
  <table>
        <tr>
                <th></th>
                <th scope="col">Saturday</th>
                <th scope="col">Sunday</th>
        </tr>
        <tr>
                <th scope="row">Tickets sold:</th>
                <td>120</td>
                <td>135</td>
        </tr>
        <tr>
                <th scope="row">Total sales:</th>
                <td>$600</td>
                <td>$675</td>
        </tr>
  </table>
</body></html>
```

Table Headings, example

	Saturday	Sunday
Tickets sold:	120	135
Total sales:	$600	$675

Figure 10: Defining a table and using table headings

4.1.9 Site Map

- Offers a visual representation of the information space in order to help users understand where they can go.
- Give users an overview of the site's areas in a single glance.

5 Lecture 2b: URLs and Site Structures

- URL: Uniform Resource Locator.
- Specifies the precise location of a resource on the Internet.
- A web browser communicates with web servers using the HyperText Transfer Protocol (HTTP). Therefore, the URLs for all web pages must start with the scheme `http` or `https` (for secured data transmission)

```
http://www.example.com/path/to/myfile.html?key1=value1#SomewhereInTheDocument
```

- `http://` is the protocol part of the URL. Specifies which protocol to use. Other possible protocoles: `mailto:` (opening a mail client), `ftp:` (for handling file transfer)
- `www.example.com` Domain name. Indicates which web server is being requested. Also possible to use an IP address directly.
- `/path/to/myfile.html` Path to the resource on the web server.
- `?key1=value1` extra parameters provided to the web server. A list of key/value pairs, separated with the `&` symbol.

- `#SomewhereInDocument` is an anchor to another part of the resource itself.

GET vs. POST

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

Figure 11: Difference on using GET or POST as a method for URLs

6 Lecture 2c: Introduction to CSS

- CSS: "Cascading Style Sheets". Used for separating structure (HTML) from appearance (CSS).
- CSS sees HTML as boxes:

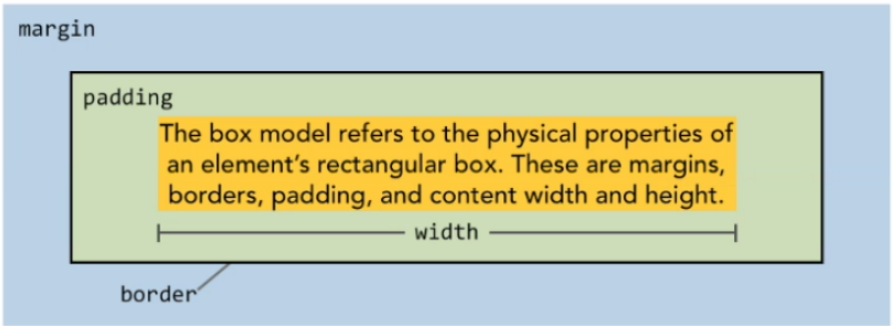

Figure 12: The box model for CSS.

- CSS syntax:

```
p { // p is the selector -> "All paragraphs"
    // "property: value"
    font-family: Arial, sans-serif;
    font-size: 16px;
    color: black;

```

```
7 }
```

- Three ways of specifying styles:
  - External style sheet (multiple pages can link to this)
  - Internal/embedded style sheet (applies only to specific web page).
  - Inline style (applies only to the specific element)

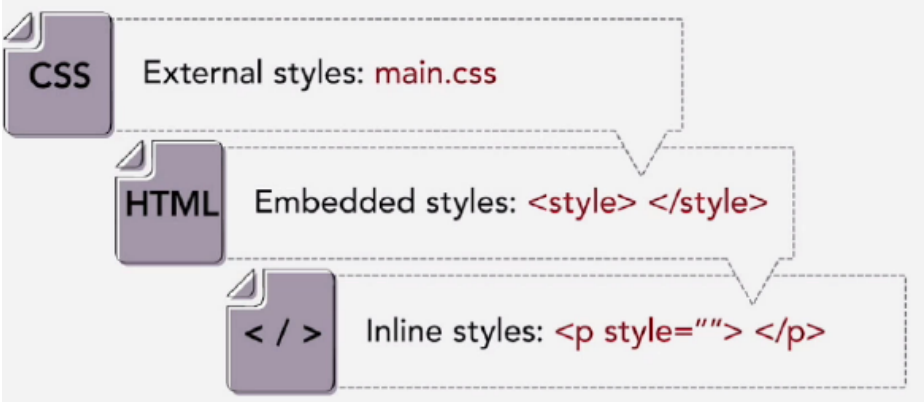

Figure 13: How to apply CSS style to HTML documents.

6.1 Inline Styles

Examples:

```
1 <body style="background-color: yellow">
2
3 <h1 style="text-align: center">
4
5 <p style="font-family: courier new; color: red; font-size: 20px;">
```

- Not maintainable for large HTML pages.

6.2 Embedded Styles

- Specify styles for each element in the given page. Specified in the head of the HTML document:

```
1 <head>
2     <style type="text/css">
3         body {
4             font-family: Helvetica, Arial, sans-serif;
5             color: #665544;
6             padding: 22px;
7         }
8     </style>
9 </head>
```

6.3 External Style Sheet

- Style rules stored in a separate file (e.g. `mystyle.css`)
- Each page wishing to use this style links to the external file:

```
1 <head>
2     <link rel="stylesheet" type="text/css" href="mystyle.css" />
3 </head>
```

6.4 The concept of Cascading

- Three main sources of style information form a *cascade*:
  - The browser's default styles for the markup language
  - The styles linked to the document by its author. These can be specified, in three places, as above: Internally, embedded or externally
  - Styles specified by a user who is reading the document.

Resolving rule conflict

- Rules in user style sheets have higher priority than author style rules. Use `!important` to override: `p color: blue !important`
- All user rules and author rules have more weight than the default supplied by the web browser.
- Later styles over-rules earlier styles:

```
<style type="text/css">
        body {
                background-color:yellow;
                font-weight:bold;}
        div {
                background-color:#afa;
                font-weight:normal;}
</style>
    <p>Some text here, inherits properties of the body.</p>
    <div>
    <p>However, the div's rules over-ride the body's rules, as the div's rules apply
    later (i.e. nearer to this text in the document).</p>
    </div>
```

Figure 14: The latest applied styles overrule others.

- Style prioritization for one document:
  `Inline > Embedded > External`
  The more local a style is, the higher priority it has
- Elements can inherit styles from its parents:

This text will be in a serif font.

Now the paragraph also has a red border.
Border properties are not inherited by children, by default, but because I set border to "inherit", it is.

Figure 15: Inheriting styles from the parent element.

- The HTML `<span>` tag is used to mark sub-sequences of text:

```
<p>
    I have <span style="color: brown;">brown</span> eyes and my wife
        has <span style="color:blue;">blue</span> eyes.
</p>
```

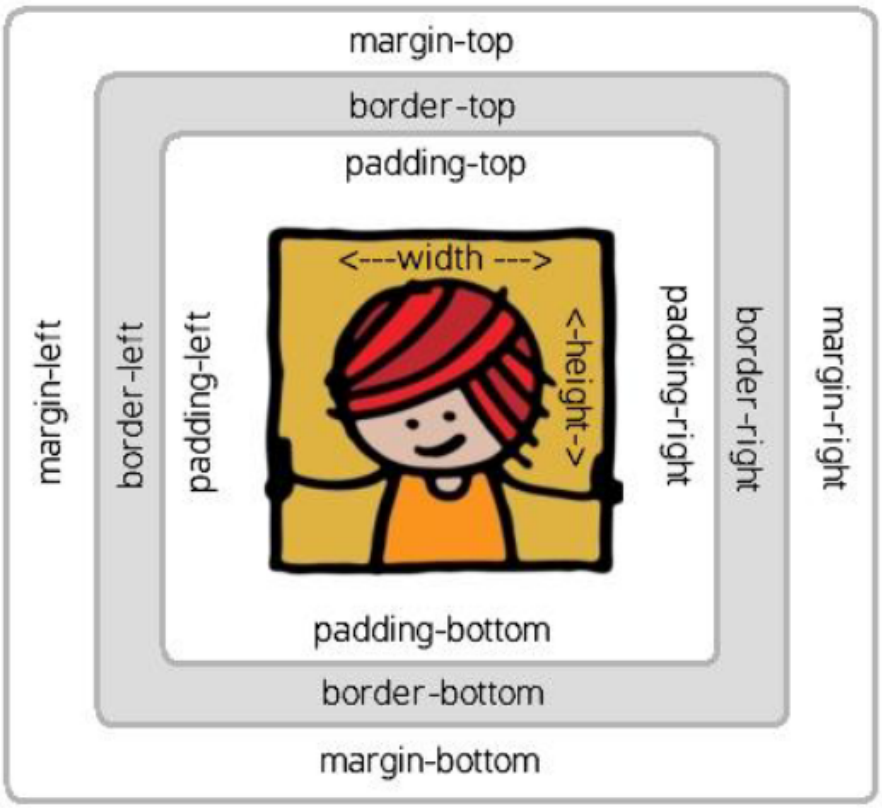

Figure 16: Styling the dimensions of a CSS box.

7 Lecture 3a: CSS selectors

For the following HTML document, the corresponding HTML document tree is shown in Figure 17

```
<html>
    <head>
        <style type="text/css">
        </style>
    </head>
    <body>
        <h1>Heading 1</h1>
        <h2>Heading 2</h2>
        <p>First paragraph.</p>
        <p>
            Second paragraph has a <b>bold</b> and a <span>span with
               another <b>bold</b></span>
        </p>
    </body>
</html>
```

This leads to the following document tree:

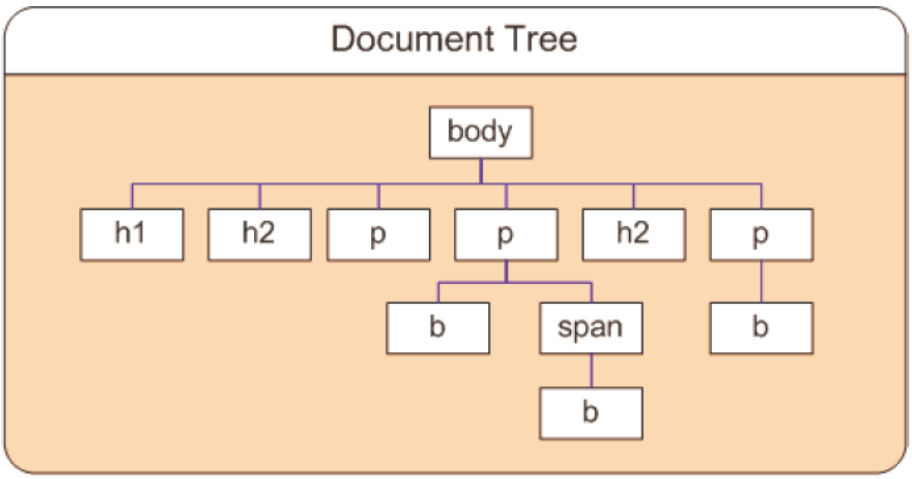

Figure 17: Corresponding document tree.

7.1 Grouping

Several selectors may be grouped together by a comma-separated list:

```
h1 {
    font-family: sans-serif;
}
h2 {
    font-family: sans-serif;
}
h3 {
    font-family: sans-serif;
}
// is equivalent to
h1, h2, h3 {
    font-family: sans-serif;
```

```
}
```

7.2 Universal Selector

Selects every element on the page

```
* {
    background-color: red;
}
```

7.3 Class Selector

Selects an element that matches a class name defined in a class attribute in the HTML.

```
.element {
    background-color: red;
}
```

7.4 ID Selector

Selects an element that matches the ID defined in an `id` attribute in the HTML

```
#elementID {
    background-color: red;
}
```

7.5 Descendant selector

Defined with a space character, separating two selectors. Represents a child element, but not just immediate children, but further nested ones as well.

```
p a {
    background-color: red;
}
```

This will target any `<a>` elements that sit inside a `<p>` element, even if there are no elements nested between them.

7.6 Attribute Selector

Targets an element based on a HTML attribute and/or attribute value.

```
div[style] { // Targets any <div> element that has a "style" attribute.
    background-color: red;
}

input[type="text"] { // Targets any <input> element that has a "type"
    attribute with a value of "text".
    background-color: red;
}
```

7.7 Child Selector

Selects an element based on it being an immediate child of another element.

```
1 p > b {
2     background-color: red;
3 }
```

This will thus not style a `<b>` element unless it is an immediate child of a `<p>` element. Can not be nested. Has to be an immediate child. Only the child will be targeted, not the parent.

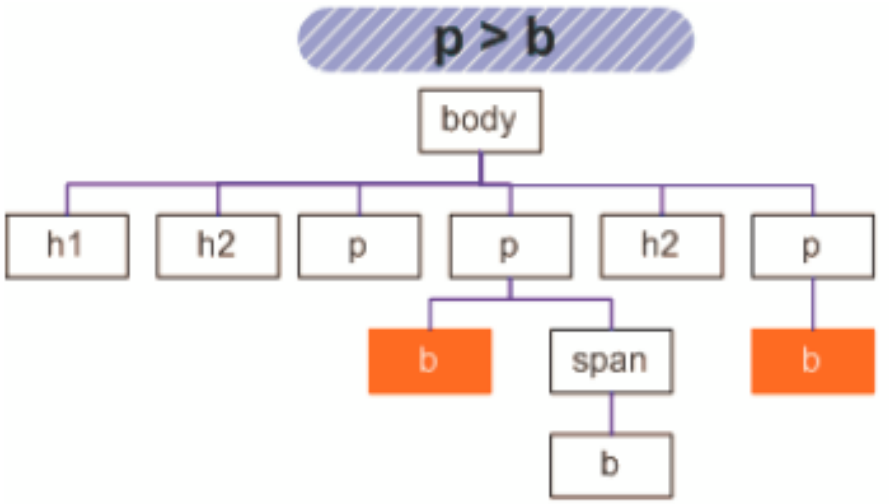

Figure 18: Example of the child selector.

7.8 Adjacent Sibling Selector

Targets elements that are "adjacent" to each other, or immediate siblings. They must have the same parent element.

```
1 h2 + p {
2     background-color: red;
3 }
```

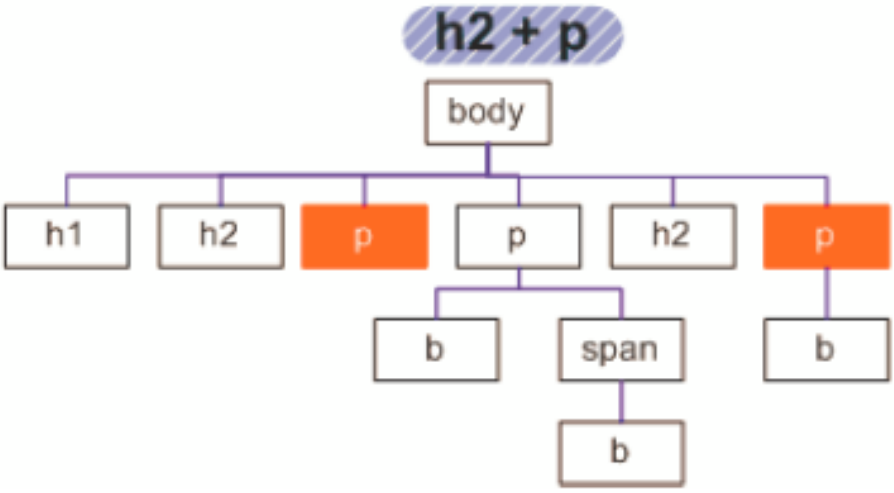

Figure 19: The adjacent sibling.

7.9 General Sibling Selector

The same as the adjacent sibling selector, except that the elements do not have to be immediate siblings.

```
1 h2~p {
2     background-color: red;
3 }
```

If you have to `<p>` elements that are siblings of a `<h2>` element, this rule would apply to both.

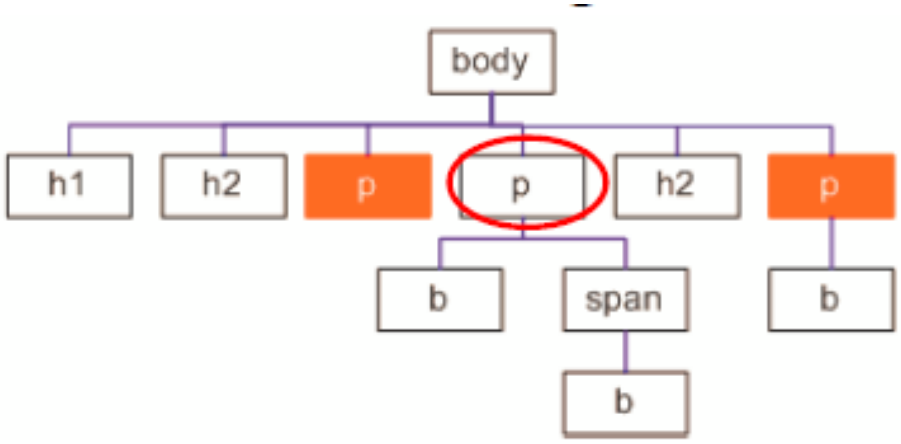

Figure 20: The General siblings

7.10 Pseudo-class

Selects an element based on the state it is in.

```
1 a:visited {
2     background-color: red;
3 }
4
5 a:hover {
```

```
6       background-color: red;
7   }
```

7.11 Pseudo-element

Adds special effects to some selectors. Represent elements in the HTML page that are not really a part of the rendered HTML

```
1   p::first-letter {
2       background-color: red;
3   }
4
5   p::before {
6       content: "Read this-";
7       background-color: yellow;
8   }
```

Note on double-colons: A part of CSS3. Attemps to distinguish between pseudo-classes and pseudo-elements. Pseudo-elements create new virtual elements.

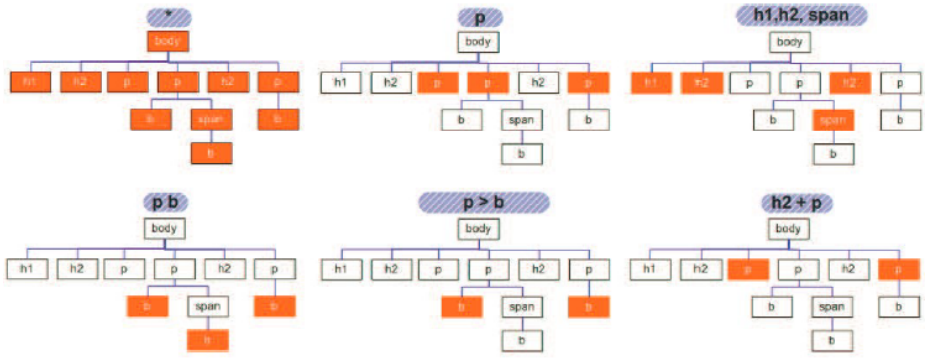

Figure 21: The most common selectors summarized.

8 Lecture 3a: CSS Properties

8.1 Color Properties

- It is possible to specify the foreground color and background of an element. Background may be colors or images.

```
1   h1 {
2       background-color: red;
3   }
4   body {
5       background-image:URL("stripe.gif");
6   }
```

- Color values can be specified by a named color: `background-color: red;` or a numerical RGB specification: `background-color: rgb(255, 0, 0)`, or `background-color: rgb(100%, 0%, 0%`

8.2 Font Properties

- The font-family specifies the order of preference:

```
1  body {
2      font-family: "Book Antigua", "Times New Roman", serif;
3  }
```

- Font style, variant and weigth is also possible to set:

```
1  h1, h2, h3 {
2      font-style: italic;
3  }
4  h3 {
5      font-variant: small-caps;
6  }
7
8  strong {
9      font-weight: normal;
10 }
```

8.2.1 Font Size

- Five standard units are used: mm, cm, in, pt, pc
- Relative font size makes web pages able to dynamically adapt to the correct font.
- For this, the *percentage* and *em* unit is used.

```
1  p {
2      font-size: 150%;
3  }
4  em {
5      font-size: 1.5em;
6  }
```

- 100% or `1em` is equal to font size of the **parent** element.

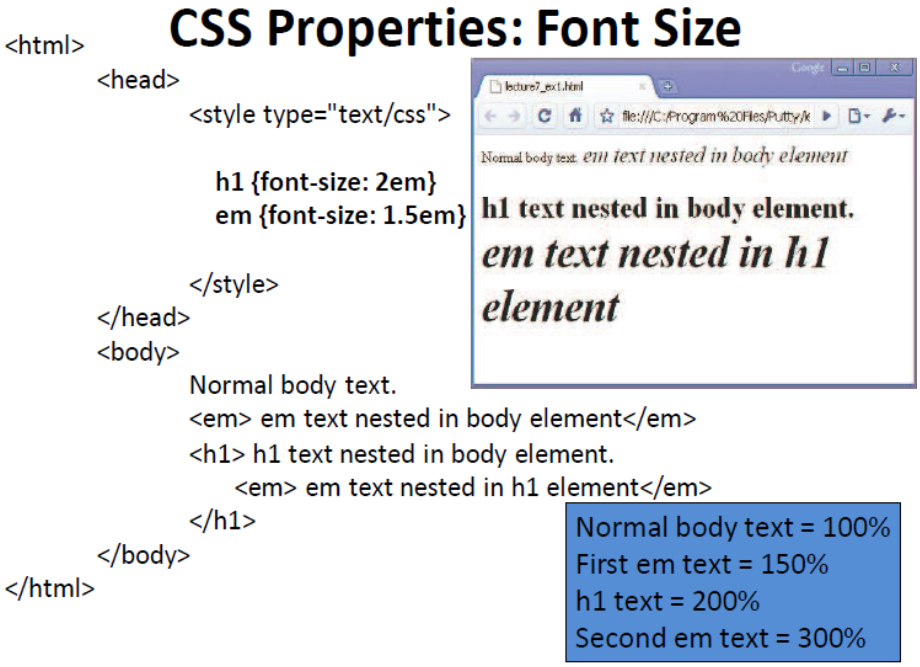

Figure 22: Relative font sizes explained.

- Font size can also be specified in pixels. 1 pixel = 1 dot on output device.

8.3 Text Properties

- Presentation of text can be adjusted by
  - Indentation:

```
p {
    text-indent: 3em;
}
```

  - Text alignment: `left, center, right, justify`

```
div.center {
    text-align: center;
}
```

  - Text decoration: `none, underline, overline, line-through`

```
a[href] {
    text-decoration: underline;
}
```

8.4 Box Properties

- CSS box model is a rectangle that wraps around every HTML element. Composed of margin, border, padding and content.

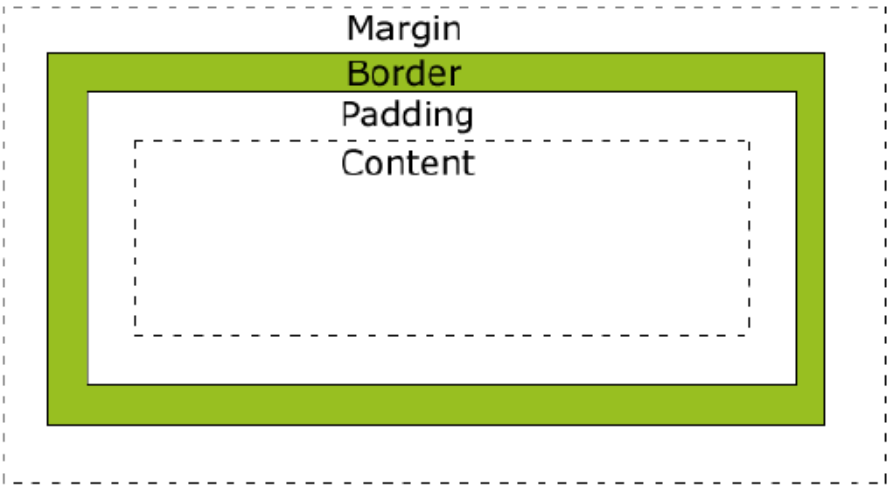

Figure 23: The CSS box model

- Margin: Clears an area around the border. Does not have a background color, it is completely transparent.
- Border: A border that goes around the padding and content. Inherited from the color property of the box.
- Padding: Clears an area around the content. Padding is effected by the background color of the box.
- Content: Content of the box, where text and images appear.

8.4.1 Overflowing content

- The `overflow` property tells the browser what to do if the content is larger than the box itself.

```
1 p.one {
2     overflow: hidden; // simply hides any extra text
3 }
4 p.two {
5     overflow: scroll; // Adds a scrollbar to the box.
6 }
```

8.4.2 Display styles

- The display property allows you to turn an inline element into a block element, and vice versa.

```
<ul>
        <li>Home</li>
        <li>Products</li>
        <li class="coming-soon">Services</li>
        <li>About</li>
        <li>Contact</li>
</ul>
```

Home Products About Contact

```
li {display: inline;
   margin-right: 10px;}
li.coming-soon {display: none;}
```

Figure 24: Setting display: none;

- Visibility will hide the box from the user, but leaves a space where the box should have been.

```
<ul>
        <li>Home</li>
        <li>Products</li>
        <li class="coming-soon">Services</li>
        <li>About</li>
        <li>Contact</li>
</ul>
```

Home Products About Contact

```
li {
        display: inline;
        margin-right: 10px;}
li.coming-soon {
        visibility: hidden;}
```

Figure 25: Setting visibility: hidden.

8.4.3 Rounded corners

- Rounded corners on the box can be created by `border-radius`. Complex shapes can also be created by specifying different distances for the horizontal and vertical parts of the rounded corners.

```
p.one {border-top-left-radius: 80px 50px;
p.two {border-radius: 1em 4em 1em 4em / 2em 1em 2em 1em; horiz values / vertic.
p.three {padding: 0px; border-radius: 100px;}
```

```
<p class="one"></p>
<p class="two"></p>
<p class="three"></p>
```

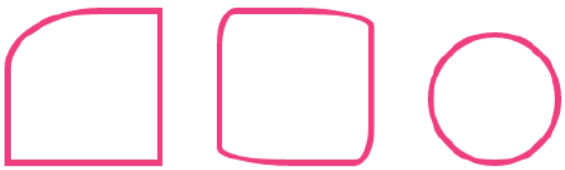

Figure 26: Making complex shapes of the borders.

8.5 Positioning

8.5.1 position: static

- Each block-level element sits on top of the next one. The **default** way that browser treat HTML.

8.5.2 position: relative

- Moves an element in relation to where it would have been in normal flow.

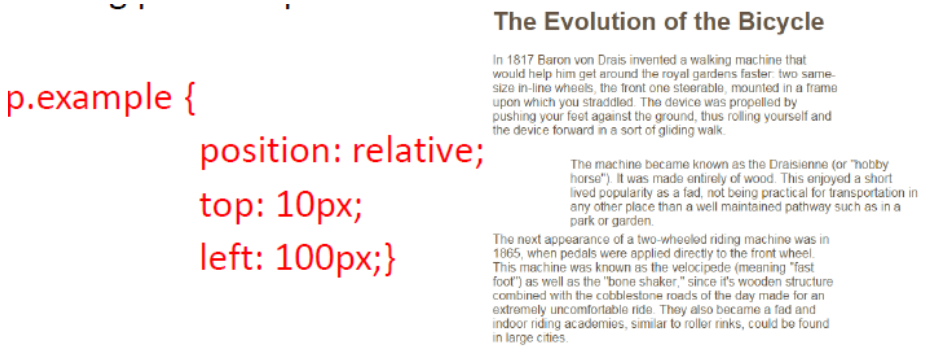

Figure 27: Relative positioning

8.5.3 position: absolute

- The box is taken out of normal flow and no longer affects the position of the other elements.

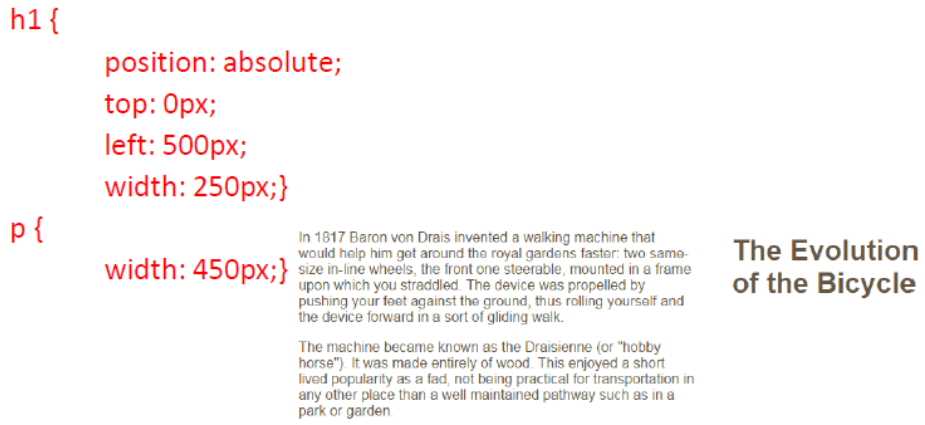

Figure 28: Absolute positioning

8.5.4 position: fixed

- The position is specified in relation to the browser window. As such, the user scrolls down, but the element stays on the same place.

Property Value	Description
static	This is the default setting – no special positioning.
absolute	Move element relative to upper left corner of page or a containing element.
relative	Move element relative to its default position.
fixed	Move element relative to browser window – ie doesn't change position if scrolling content.

Figure 29: Positioning summarized.

- Overlapping elements are solved with the `z-index` property. The higher value the `z-index` property has, the closer that element is to the front.

8.5.5 Floating elements

- Allows you to take an element in normal flow and place it as far as possible to the left or right.

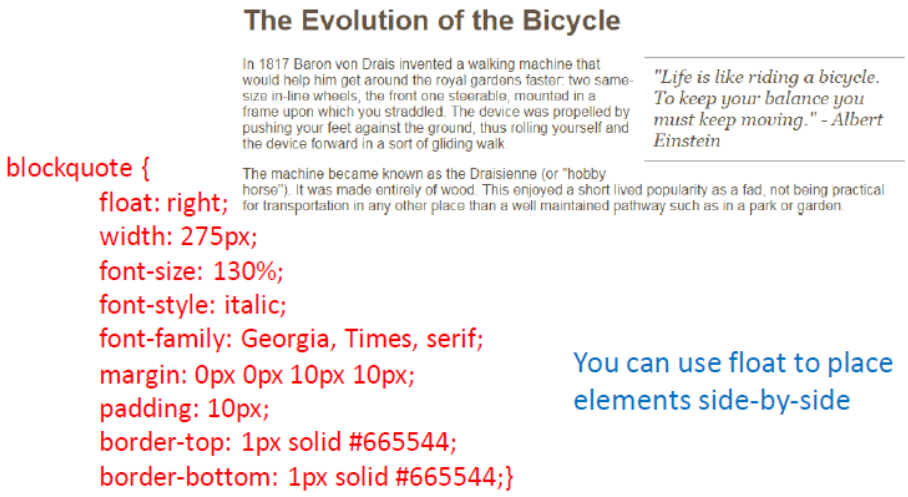

Figure 30: Floating to the right.

8.6 Table properties

- Most used properties are:
  - Width
  - Padding: Set the space between the border of each cell.
  - Letter-spacing, font-style: Set additional styles.
  - `border-top, border-bottom`: Set a border below and above headers
  - `:hover` used to highlight a table row when a user's mouse goes over it.

```
<table><tr>
        <th>Author</th>
        <th>Title</th>
        <th class="money">Reserve Price</th>
        <th class="money">Current Bid</th>
</tr><tr>
        <td>E.E. Cummings</td>
        <td>Tulips & Chimneys</td>
        <td class="money">$2,000.00</td>
        <td class="money">$2,642.50</td>
</table>
```

```
body {
        font-family: Arial, Verdana, sans-serif;
        color: #111111;}
table {  width: 600px;}
th, td {  padding: 7px 10px 10px 10px;}
th {
        text-transform: uppercase;
        letter-spacing: 0.1em;
        font-size: 90%;
        border-bottom: 2px solid #111111;
        border-top: 1px solid #999;
        text-align: left;}
tr.even {  background-color: #efefef;}
r:hover {  background-color: #c3e6e5;}
.money {  text-align: right;}
```

First Edition Auctions

AUTHOR	TITLE	RESERVE PRICE	CURRENT BID
E.E. Cummings	Tulips & Chimneys	$2,000.00	$2,642.50
Charles d'Orleans	Poemes		$5,866.00
T.S. Eliot	Poems 1909 - 1925	$1,250.00	$8,499.35
Sylvia Plath	The Colossus		$1031.72

Figure 31: Example of used table properties

9 Lecture 4: Working with Multimedia on the Web

- Most browser suppurt the major image file formats:
  - GIF
  - JPEG
  - PNG
  - SVG

9.1 GIF: Graphics Interchange Format

- Limited to displaying 256 colors.
- GIF works best on images with few colors, such as blocks of single colors, icons, buttons, logos, clip art and line drawings (tracings).

9.1.1 Interlaced and non-interlaced

- **Non-interlaced:** The image appears as it is slowly retrieved by the web browser.
- **Interlaced:** The graphic starts out as a blurry representation of the final image. Then gradually comes into focus.

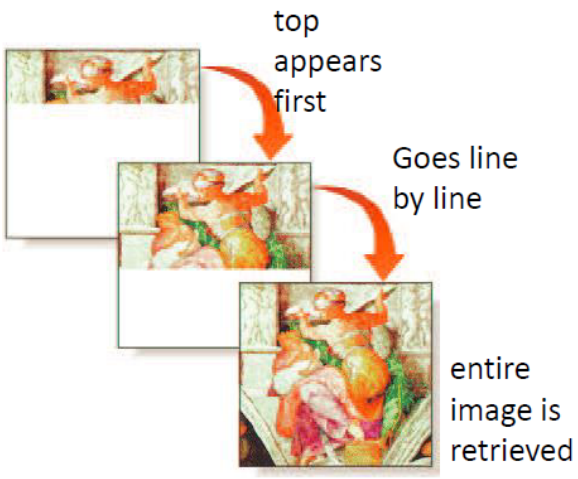

Figure 32: Non-interlaced image

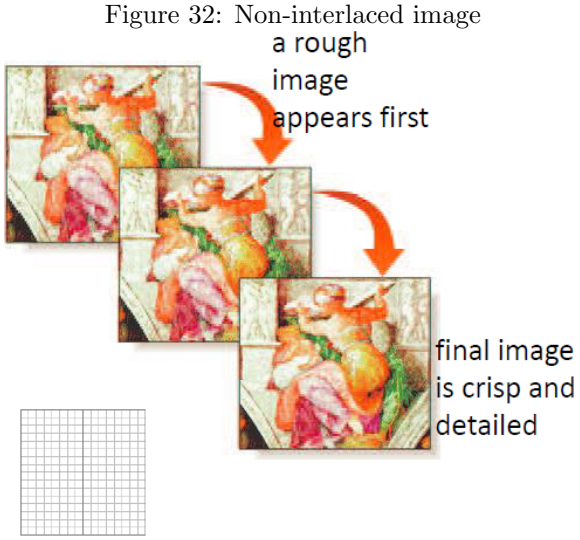

Figure 33: Interlaced images

- GIFs also give the possibility to be saved as transparent, and to make animations, where several images are displayed in rapid succession.

9.2 JPEG, Joint Photographic Experts Group Format

- Designed for storing full-color or greyscale images or real life (photography).
- Stores 24-bit color pictures
- Lossy compression with compression ratio of about 10:1 or 20:1. Best lossless compressions: $\approx 2:1$.

9.3 PNG: Portable Network Graphics

- A sort of improved version of GIF
- Uses a free algorithm for compression.
- Supports 8-bit transparency. Allows for the illution of smooth curves, and makes curved images look good against any background.

9.4 SVG: Scalable Vector Graphics

- A vector graphics format
- Can be scaled independently of resolution.

9.5 Image map

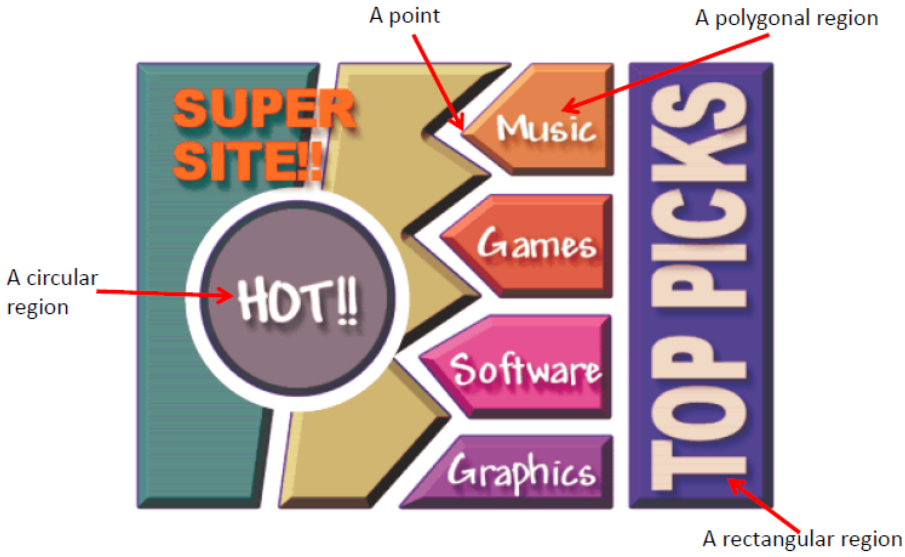

Figure 34: Example of an image map

- Made up of an image and a list of hotspots.
- Syntax of hotspot element: `<area shape=\shape" coords =\coordinates" href =\url" alt=\text" />`
- Sample code for a rectangular hotspot:

```
1 <area shape="rect" coord="384, 61, 499, 271" href="water.html" \>
```

9.6 Sound formats

- WAV, Waveform Audio File Format: Usually contains uncompressed audio. High quality but large file sizes.
- MP3: Compresses audio files with minimal impact on sound quality.
- WMA, Windows Media Audio: uses Microsoft's compression algorithm.
- Ogg: Supported by HTML5
- MIDI, Musical Instrument Digital Interface: Limited to instrumental music and not speech/-general sounds.
- Adding HTML5 audio to your page:

```
<audio controls autoplay>
    <source src="audio/test-audio.ogg" />
    <source src="audio/test-audio.mp3" />
    <p>This browser does not support our audio format.</p>
</audio>
```

9.7 Video files

- Video file formats supported by HTML5:
  - .mp4
  - .ogg
- Adding a video element:

```
<video src="puppy.mp4"
    poster="puppy.jpg"
    width="400"
    height="300"
    preload="none"
    controls
    loop
    >
</video>
```

- Multiple `<source>` elements can be used inside the `<video>` element.

10 Lecture 5: HTML Forms

10.1 The form element

- A form element is structured as follows:

```
<form attributes>
    elements
</form>
```

- `attributes` are the attributes that control how the form is processed
- `elements` are the elements placed within the form.
- The `<form>` element has two attributes:
  - `method` indicates how the data is transmitted to the server using HTTP.
    - GET: the form data is appended to the URL (not secure)
    - POST: the form data is sent as a separate message
  - `action`: What to do when the input type `submit` is pressed.

  Example of a form:

```
<form action="forms/exForm.html" method="post">
    <fieldset>
        <legend>Your review</legend>
        <label for="hear-about">How did you hear about us?</label>
        <select name="referrer" id="hear-about">
            <option value="google">Google</option>
            <option value="friend">Friend</option>
            <option value="advert">Advert</option>
            <option value="other">Other</option>
        </select>
        <p>Would you visit us again?</p><br/>
        <label><input type="radio" name="rating" value="yes">Yes</
            label>
        <label><input type="radio" name="rating" value="no">No</
            label>
        <label><input type="radio" name="rating" value="maybe">Maybe
            </label>

        <label for="comments">Comments</label><br/>
        <textarea rows="4" cols="40" id="comments"></textarea>
        <label><input type="checkbox" name="subscribe" checked="
            checked"/>Sign me up for email updates</label><br/>
        <input type="submit" value="Submit review"/>
    </fieldset>
</form>
```

Figure 35: The generated example form

11 Lecture 5b: Using Inspector and Responsive Web Design

The eight rules of usability:

- Strive for consistency
- Cater to universal usability
- Offer informative feedback. "Has the form been submitted?"
- Design dialogs to yield closure
- Prevent errors
- Prevent easy reversal of actions. Don't let a user get lost on your website. Make it easy for them to go back to whatever page they were on.
- Support internal locus of control. Make sure your users always know where on your website they are located.
- Reduce short term memory. Do not make your users remember information from page to page.

11.1 Responsive Web Design

- *Designing your website so that it is adaptable and accessible to users on any device.*
- Guidelines for designing for smaller screens:
  - Resize you content to fit the screen. If your content does not fit horizontally, stack it vertically.
  - Remove non-essential content.
  - Increase font-size for legibility.
  - Make your links and buttons recognizable and clickable.
  - Whitespace is still important.
- Responsive design is often done with a media query. A media query is like an if statement: if your screen is within a certain range, then apply the CSS rules. Example of a media query:

```
.colorblock {
    width: 100%;
    max-width: 850px;
    height: 30px;
    background-color: red;
}

@media screen and (max-width: 768px) {
    .colorblock {
        background-color: blue;
    }
}

@media screen and (max-width: 480px) {
    .colorblock {
        background-color: green;
    }
}
```

12 Lecture 6: Introduction to JavaScript

- JavaScript is used to dynamically create content. Because user interaction requires web page content to change.
- HTML is for content. CSS is for styling/presentation. JavaScript is for controlling the behaviour.
- The web browser first look at the content (the HTML). Then applies the style (CSS). Lastly, it interprets the JavaScript.
- JavaScript files should be linked right before the closing `</body>` tag. This is because the browser will execute the JavaScript when it comes across a `<script>` tag. This blocks further content loading until it finishes executing the JavaScript. For example, if the JavaScript needs an element inside the content before the content is loaded, will cause the site loading to crash.

```
<!DOCTYPE html>
<html>
<head>
    <title>My Title</title>
</head>
<body>
    <p>Content</p>
    <script src="myScript.js"></script>
</body>
</html>
```

12.1 Variables

- JS has dynamic data types. Meaning that the same variable can be used as different types (`int, bool, string, ...`
- Variables are declared using the keyword `var`. In newer browser, keywords `const` and `let` is also possible.
- `const` defines a constant that can not be changed.
- `var` means that the variables will be function scoped. Only accessible inside the function it is declared in, and will be global within that scope.
- `let` declares a variable that are limited in scope of the block.

```
function varTest() {
  var x = 1;
  if (true) {
    var x = 2;  // same variable!
    console.log(x);  // 2
  }
  console.log(x);  // 2
}

function letTest() {
  let x = 1;
  if (true) {
    let x = 2;  // different variable
    console.log(x);  // 2
  }
```

```
16    console.log(x);  // 1
17 }
```

- As we have dynamic variables, it is not needed to define its data type:

```
1 var text = 'javascript'      // string
2 var truefalse = true         // boolean
3 var myNum = 42               // number (int)
4 var someNum = 2.7            // number (float)
5 someNum = false              // type changed. Now boolean
6 list = []                    // array
```

- Types can be checked using `console.log(typeof myNum)` would print `number`
- JavaScript is case-sensitive.

12.2 Parsing string to number

- When you want to work on numbers, but your number is a string.

```
1 var myStringNumber = '123';          // Returns '123'
2 // Parse to a real number:
3 var myNumber = parseInt(myStringNumber);          // Returns 123
```

12.3 Functions

- Functions are "first class citizens". Meaning that it will allow you to pass funcitons as parameters, return a function from a function call, assign a function to a variable, or storing them inside an object/other data structures. Defining functions:

```
1 // Method 1:
2 function functionName(params) {
3     ...
4 }
5
6 // Method 2:
7 var functionName = function(parameters) {
8     ...
9 }
```

- The variables in the functions are local. When you declare a variable withn a function, the variable can only be accessed within that function:

```
1 var functionName = function(params) {
2     var a = 'JavaScript'
3     console.log(a)          // writes 'JavaScript' to the console.
4 }
5 // a is not accessible outside the function.
```

12.4 `if/else` sentences

```
var a = 'it2805'
if (a == 'it2805'){
    ...
} else {
    ...
}
```

- `if`: execute some code if the condition is true.
- `if else`: execute some code if the condition is true, else execute something else if the condition is false.
- `if ... if else ... else`: select one block of code to execute.

12.5 `switch` cases

```
switch (a) {
    case 'it2805':
        ...
        break
    case 'it2810':
        ...
        break
    default:
        ...
}
```

```
var myInput = 'webtech';

switch (myInput) {
  // single case
  case 'webtech':
    console.log('this is awesome!');
    break;

  // multiple cases
  case 'world':
  case 'hello':
    console.log('Hello, world!');
    break;

  // default execution of none of the conditions aboce were met
  default:
    console.log('did\'t match any of the cases');
}
```

12.6 `for` loop

```
for ([initialization]; [condition]; [final-expression]) {
    statement
}
```

- `initialization` is an expression evaluated before the program loop.
- `condition` is an expression evaluated at the start of each loop. The loop is terminated if the test evaluates to false.
- `final-expression` is an expression that is evaluated at the end of each loop.

12.7 The `while` loop

```
1 while (condition) {
2     statements
3 }
```

12.8 Operators

12.8.1 Arithmetic operators

`+, -, *, /, %, ++, --`

```
1 var a = 1;
2 var b = 5;
3 a++;          // a = 2
4 b--;          // b = 4
```

12.8.2 Assignment operators

`=, +=, *=, /=, %=`

```
1 var a = 1;
2 a += 5        // a = a + 5 = 1 + 5 = 6
```

12.8.3 Logical operators

`&&, ||, !` corresponding to AND, OR, NOT

12.8.4 Comparison operators

`==, ===, !=, !==, >, <, <=, >=`

- `==`: equality
- `===`: equality in value and type
- `!=`: not equal
- `!==`: not equal in value and type

12.9 The DOM: Document Object Model

- The programming interface for HTML, XML and SVG documents.
- Lets us perform actions and manipulations on the web page (the DOM).

```
1 // Getting an element from HTML by ID:
2 var el = document.getElementById('myElementId');
3
4 // Updating the content of the element:
5 el.innerHTML = 'New content';
```

13 Lecture 7: XML, JSON, DOM, Events

13.1 XML, eXtensible Markup Language

- A markup language specifies the structure and content of a document.
- There is no connection between HTML and XML other than that they have both been defined using SGML, Standard Generalized Markup Language.
- XML is a great data-interchange format.
- A **well formed** XML document does not contain any syntax errors.
- A **valid** XML document is a well formed document, but also satisifies the rules laid out in the DTD or schema attached to the document.
- If an XML document contains errors, the parser will stop reading it.
- XML documents consist of three parts:
  - **The prolog** is optional and provides information about the document.
    Consists of four parts in the following order:
    * XML declaration
    * Miscellaneous statements or comments
    * Processing instructions
    * Document type declaration

    ```
    1 // The format:
    2 <?xml version="version.number" encoding="encoding type"
          standalone="yes|no" ?>
    3
    4 // Example:
    5 <?xml version="1.0" encoding="utf-8" standalone="yes|no" ?>
    ```

  - The body contains the content of the document.
  - The epilog is also optional and contains any final comments or processing instructions.

13.1.1 Namespacing

- Used to avoid name collisions
- Prefixed names are called **qualified names**
- An element name *without* a namespace prefix is called an **unqualified name**.

```
<!-- Example of a name collision -->
<model>
    <!-- HTML table -->
    <table>
        <tr>
            <td>This is content.</td>
        </tr>
    </table>

    <!-- a furniture -->
    <table>
        <material>Jarrah</material>
        <cost>$100</cost>
    </table>
</model>
```

- This collision can be avoided using namespacing. A namespace is declared in the prolog/root using `xmlns:prefix="URI"`

```
<model>
    xmlns:a="http://w3.org/TR/html4"
    xmlns:b="http://idi.ntnu.no/furniture"

    <!-- HTML table -->
    <a:table>
        <a:tr>
            <a:td>This is content.</a:td>
        </a:tr>
    </a:table>

    <!-- a furniture -->
    <b:table>
        <b:material>Jarrah</b:material>
        <b:cost>$100</b:cost>
    </b:table>
</model>
```

- It is also possible to declare it in the Element section:

```
<model>
    xmlns:a="http://w3.org/TR/html4"
    xmlns:b="http://idi.ntnu.no/furniture"

    <!-- HTML table -->
    <a:table xmlns:a="http://w3.org/TR/html4">
        <a:tr>
            <a:td>This is content.</a:td>
        </a:tr>
    </a:table>

    <!-- a furniture -->
    <b:table xmlns:b="http://idi.ntnu.no/furniture">
        <b:material>Jarrah</b:material>
```

```
15          <b:cost>$100</b:cost>
16      </b:table>
17  </model>
```

13.1.2 Stylesheet

- It is possible to style XML using CSS.
- Linking a stylesheet to your XML document:

```
1  <?xml-stylesheet type="text/css" href="style.css" ?>
```

13.2 JSON, JavaScript Object Notation

Example of JSON:

```
1  {
2      "root": {
3          "children": [
4              {"name": "Per"},
5              {"name": "Lars"}
6          ]
7      }
8  }
```

13.2.1 Structures

- **Values** can be `string`, `number`, `object`, `array`, `booleans` and `null`.
- **Objects** are sets of unordered name/value pairs:

```
1  {name1: value1},
2  {name2: value2}
```

- An array is an ordered set of values: `[val1, val2, val3]`

13.3 JavaScript

13.3.1 Objects

- Initializing an object:

```
1  var myObject = {
2      myProperty: 'value1'
3      anotherProperty: 'value2';
4  }
5
6  // Accessing object properties is done like this:
7  myObject.myProperty;         // this returns 'value1
8
9  // Updating object properties:
10 myObject.myProperty = 'newValue';
```

- Properties can also be functions:

```
var myCar = {
    speed: 100,
    accelerate: function(){
        this.speed = this.speed + 40;
    }
};
myCar.accelerate();              // will increase speed with 40.
```

13.3.2 Objects using constructor functions

- A constructor lets you create multiple instances/objects of the same function:

```
var Person = function (name, age) {
    this.name = name;
    this.age = age;
}
var Peder = new Person('Peder', 12);
var Espen = new Person('Espen', 13);
```

13.4 The DOM

- The DOM connects web pages to "you programming language"

13.4.1 Events

- An event handler tells the browser what code to run in response to a specified event.

```
// HTML:
<input type="text" id="name">

// JavaScript:
var input = document.getElementById('name');

input.addEventListener('keydown', function(e)) {
    e.target.value = e.target.value.toUpperase()
}
```

- Change is when an element is activated/changed. (Radio buttons, checkboxes, datepicker...)
  Example of change:

```
// HTML:
<input type="text" id="name">

// JavaScript:
var input = document.getElementById('name');

input.addEventListener('change', function(e)) {
    e.target.value = e.target.value.toUpperase()
}
```

- Example of 'click':

```
1 // HTML:
2 <input type="text" id="name">
3
4 // JavaScript:
5 var input = document.getElementById('name');
6
7 input.addEventListener('click', function(e)) {
8     alert('I was clicked!');
9 }
```

14 Lecture 8: DOM, Ajax, Events, and more

14.1 More about the DOM

- The DOM is a structured representation of the document as a tree.
- To access the DOM you could use the global `document` interface.

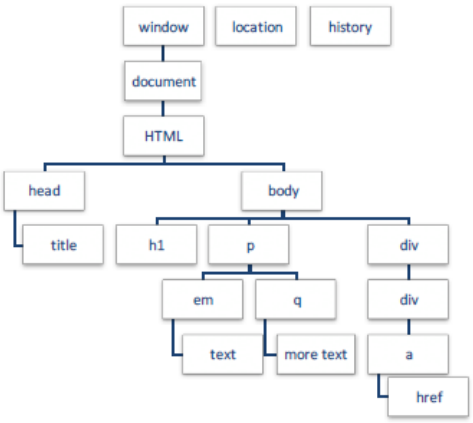

Figure 36: The DOM tree

Object name	Description
window	The browser window
document	THe web document displayed in the window
document.body	The body of the Web document
event	Events or actions occuring withint the browser window
history	List of previously visited web sites within the browser window
location	URL of the document currently displayed
navigator	The browser itself
screen	The screen displaying the document

Figure 37: Description of the DOM tree

14.2 DHTML, Dynyamic HTML

- We use the concept of DHTML for DOM manipulation.

- DHTML vs AJAX: DHTML is request/reload-based. AJAX loads data from server when needed.
- DHTML loads all data into the client, and dynamically the active content.

14.3 Changing the CSS/style through JavaScript

```
1 var el = document.getElementById('myText');
2 e.style.color = '#f00';          // changing the color to red
```

14.4 Accessing forms through JavaScript

```
1 document.fname       // using the form name
2 document.forms[index]        // using the index
3 document.forms['formID']     // using the ID
4 document.getElementById('fid')  // select by ID
```

- The `onChange` event handler can give you real time forms validation
- For preventing the default form to be sent in by mistake, you could use the `event.preventDefault()` function.
- Never trust user input. Server-side validation of a submitted form must always be the final validatation.

14.5 Ajax

- Lets web applications make quick, incremental updates to the user interface without reloading the entire browser.
- Differs from the DHTML by having the opportunity to fetch data when needed.

14.6 JSONP

- Is used to overcome cross-domain restrictions.
- Browsers wil not allow you to access resources retrieved from origins other than the one the current page is served from.

14.7 CORS, Cross-Origin Resource Sharing

- Enables secure cross-domain data transfer.

14.8 The Date object

```
1 let date = new Date();
2 var today = showDate(date)
3
4 function showDate(dateObj) { // a function that returns the
    formatted date
5     let day = dateObj.getDate();          // get day
```

```
6       let month = dateObj.getMonth() + 1;  // get month (jan=0 so add
            1)
7       let year = dateObj.getFullYear();    // get year
8
9       return day + '/' + month + '/' + year;
10  }
```

15 Exam H2015

15.1 Section A: Multiple Choice Questions

1. *The elements* `<div>` *and* `<span>` *have the following characteristics:*
   C. Elements `<div>` and `<span>` define content to be inline or block-level.

2. *In regards to the CSS box model, where is the margin property located?*
   B. Outside the box

3. *Which built-in HTML5 object is used to draw on the canvas?*
   A. `getContext`

4. *Which primitive shape is supported by* `<canvas>`*?*
   B. Rectangle

5. *While working on a JavaScript project, which function would you use to send messages to users requesting for text input?*
   B. Prompt()

6. *Which protocol is ideal for transmitting large files?*
   FTP, File Transfer Protocol

7. *In HTML tables, the number of coloumns is determined by*
   A. How many `<td>` elements are inserted within each row.

8. *If you would like visited links to be green, unvisited links to be blue, and links that the mouse is over to be red, which CSS rules will you use?*
   B.
   ```
   a:link {color: blue}
   a:visited {color: green}
   a:hover {color: red}
   ```

9. *From outside to inside, a box (block) has:*
   D. margins, border, padding

10. *The difference between margins and padding is...*
    B. Margin does not have a background color. The border and padding have color inherited from the `color` property of the box.

11. *What is the default value of the position property?*
    D. `static`

12. *Which HTML attribute specifies an alternate text for an image, if the image cannot be displayed?*
    C. `alt`

13. *Indicate whether or not the following statements are true or false:*

    A. *This is a well formed XML document:*

```
<?xml version="1.0" encoding="UTF-8"?>
<book1>
    <isbn>1234567890</isbn>
    <name>XML Master Basic >2006&lt;</name>
</book1>
```

False. The `<name>` attribute contains false characters: `>, &, ;`
CORRECTION: Solution says "True, this is a well-formed XML document"...

B. *In a hierarchical structure, each page is linked with the pages that follow and precede it in an ordered chain.*
False. This happens in a linear structure.

C. *In HTML5 `<video>` element you do no need to supply values for all attributes, e.g. control, loop; these attributes are on when they are there, and off in case they don't*
True.

D. *Pseudo-class selects an element based on a state the element is in.*
True

E. *POST requests are never cashed.*
True

F. *GET requests cannot be bookmarked.*
False.

15.2 Section B: Longer Answer Questions

1. *You're creating a music web site with the following folder and file structure*

```
music/
    index.html
    format/
        cd-used.html
        cd-new.html
    genre/
        rock/
            classic/
                doors.html
                stones.html
            punk/
                ramones.html
                sex-pistols.html
        rap/
            icecube.html
            DMX.html
```

Give the relative URL for...

A. *From cd-used.html to stones.html*
`../genre/rock/classic/stones.html`

B. *From `icecube.html` to `ramones.html`*
`../rock/punk/ramones.html`

C. *From `doors.html` to `stones.html`*
`./stones.html`

D. *From `index.html` to `DMX.html`*
`genre/rap/DMX.html`

2. *Fill in the following sentences*

A. The **GIF** image format tends to be good for drawn graphics and animation, but the **JPG** format is good for photos.

B. If you want to control which element sits on top, you use the **z-index** property. The **higher** value z-index property has, the closer that element is to the front.

C. You can concatenate **strings** together with the `+` operator.

D. To link to an external JavaScript file from HTML, you need the **src** attribute for the `<script>` element.

E. A parameter acts like a **local** variable in the body of a function

F. The IP addresses are provided by **DNS** that map symbolic names to their IP.

3. *Name each of the 4 required parts of a CSS rule and very briefly state the purpose of each part.*

```
selector {
    property: value;
}

/* Example */
p {
    color: red;
}

/*
p: selector. Meaning the rule applies to all <p> elements.
   Identifies the HTML tag to be styled.
{}: Declaration. Encapsulates property/value pairs.
color: property. Identifies the style attribute to be styled.
red: value. Specifies the value to set the property to.
*/
```

4. *Name each of the 4 layers in the TCP/IP model and very briefly state the purpose of each part.*

- **The Application Layer**. Provides applications the ability to access the services of the other layers and defines the protocols that applications use to exchange data, includes protocols such as the HyperText Transfer Protocol (HTTP) and the Domain Name System (DNS).
- **The Transport Layer**. Responsible for making sure that complete mesages are delivered end to end, using the Transmission Control Protocol (TCP) or User Datagram Protocol (UDP).
- The Network/Internet Layer is responsible for routing messages from one place to another; all routers on the internet run the IP protocol.
- **Physical layer.** This is responsible for actually translating the software message into a physical representation and putting them on the wire (or through the air in a wireless network or fiber-optic wire).

5. *Answer the following questions.*

```
var temp = 81;
var willRain = true;
var humid = (temp > 80 && willRain == true);
```

A. *What is the value of humid?*

```
humid = true
```

B. *What is the value of* ***isValid****?*

```
1 var guess = 6;
2 var isValid = (guess >= 0 && guess <= 6);
```

```
isValid = true
```

C. *What is the value of* ***sendFile****?*

```
1 var kB = 1287;
2 var tooBig = (kB > 1000);
3 var urgent = true;
4 var sendFile = (urgent=true || tooBig = false);
```

```
sendFile = true
```

6. *What is the result of the following code?*

```
1 <script type="text/JavaScript">
2     var x = 7;
3     var y = 5;
4     var z = 14;
5     x = y;       // x = 5
6     z = z % x;          // z = 14 % 5 = 4
7     alert(y + z);      // y + z = 5 + 4 = 9
8 </script>
```

7. *What is the result of the following code*

```
1 <script type="text/JavaScript">
2     var a = 1;
3     a = a + 1;       // a = 1 + 1 = 2
4     var b = "a is " + a;         // "a is 2"
5     a = 5;
6     alert(b);        // "a is 2"
7 </script>
```

15.3 Section C: Coding Related Questions

1. *Develop a web form to collect the following information: Name of the student, and his/her email address.*

```
1  <html>
2  <head>
3    <title>Section C, question 1</title>
4  </head>
5  <body>
6      <form name="studentForm" method="post" action="studentResult.
           html">
7        <p>Enter your name: </p>
8        <input type="text"/><br>
9        <p>Enter your email: </p><input type="email"/><br>
10       <input type="submit" value="Submit data"/>
```

```
11      </form>
12  </body>
13
14  </html>
```

2. *Consider the following HTML file:*

```
1   <html>
2   <head>
3       <style>
4           ...
5       </style>
6   </head>
7   <body>
8       <h1>Heading 1</h1>
9       <h2>Heading 2.a</h2>
10      <p>First paragraph.</p>
11      <p>
12        Second paragraph <b>first bold</b> and a <span>span with a <b>
              second bold</b></span>
13      </p>
14      <h2>Heading 2.b</h2>
15      <p>Third paragraph has a <b>third bold</b> also.</p>
16  </body>
17  </html>
```

A. *Build (draw) the document/HTML tree of the code above*

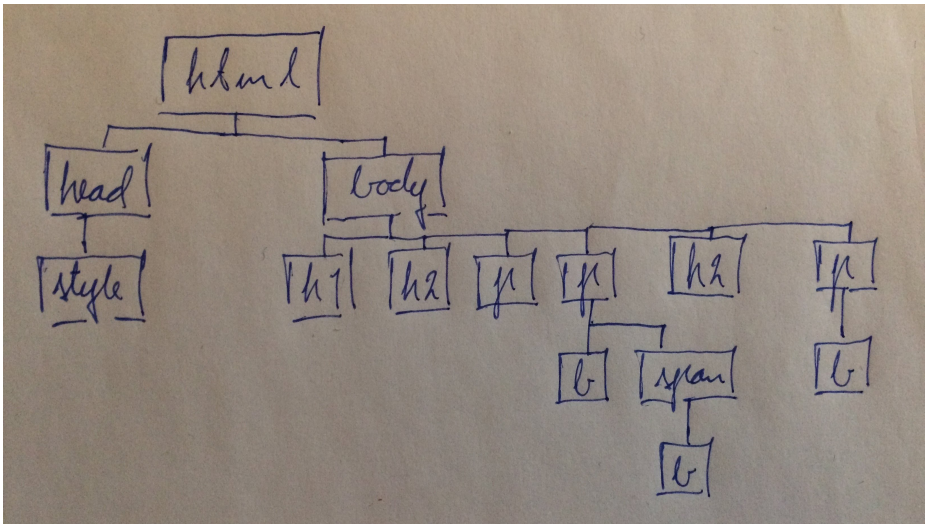

Figure 38: The corresponding HTML tree

B. *Write the CSS rules needed to produce the outcome that the words "first bold" and "third bold" should be green.*

```
1   <style>
2       p > b { /* the immediate child selector.  */
3         color: green;
4       }
5   </style>
```

C. *Write the CSS rule such that "First paragraph" and "Third paragraph has a third bold too" shall be green.*

```
<style>
    h2 + p { /* the adjacent sibling selector. */
      color: green;
    }
</style>
```

3. *Consider the following HTML*

```
<p>Lorem ipsum dolor...</p>
```

Write the CSS rules needed to

- *Change the font size for the sentence to fantasy*
- *Change its size to 24px*
- *Make the text bold*
- *Make the text italics*

```
p {
        font-family: fantasy;
        font-size: 24px;
        font-weight: bold;
        font-style: italic;
}
```

4. *Consider the following JavaScript code:*

```
var days = [Sunday, Monday, Tuesday, Wednesday, Thursday, Friday,
    Saturday];
var today = new Date().getDay();
for (var i = 1; i < 7; i++) {
    if (i = today) {
        document.write("Today is " days[today] "<br>");
    } else {
        document.write("Today is not " days[today] " "'<br>'');
    }
}
```

A. *Identify five types of errors in the code*

- Line 1: Strings must have quotes: `'Monday', ...`
- Line 2: You should use `today.getDay()`
- Line 3: `i` must start on 0 for correct indexing. (Supposing the week starts on a Sunday)
- Line 4: Must use `==` in order to get "equal in value".
- Line 5: Needs `+` sign in order to concatenate strings.
- Line 7: Can not use `''`, must use `""`

B. *Explain the effect of each error. (Done above)*

C. *Show what the corrected code would output.*

```
Today is not Sunday
Today is not Monday
Today is not Tuesday
Today is Wednesday
Today is not Thursday
Today is not Saturday
```

5. *Write some JavaScript that uses the current clock and tells whether class is over (class ends 12:20).*

```
let date = new Date();
let hour = date.getHours();
let mins = date.getMins();

let finished = false;

if (hour >= 12) {
    if (mins <20) {
        finished = false;
    } else {
        finished = true;
    }
}
```

6. *Write JavaScript code to draw the following figure inside a* `<canvas>` *element*

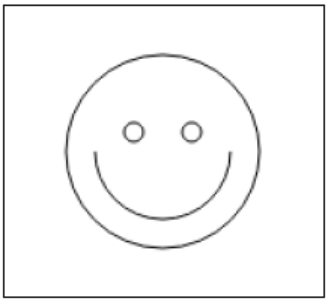

Figure 39: The smiley you should draw.

```
<!DOCTYPE html>
<html>
<head>
    <title>Section C, Q6, canvas</title>
</head>
<body>
  <canvas id="myCanvas" width="300" height="300" style="border: 1px
      solid #d3d3d3;">
  </canvas>
  <script>
// JavaScript:
var canvas = document.getElementById('myCanvas');
var ctx = canvas.getContext('2d');

```

```
14 var width = ctx.canvas.width;
15 var height = ctx.canvas.height;
16
17 // Drawing the circular face:
18 ctx.beginPath()
19 //ctx.moveTo(width / 2, height / 2);
20 ctx.arc(width / 2, height / 2, 0.8 * height / 2, 0, 2 * Math.PI,
       true);
21 ctx.stroke();
22
23 // Drawing mouth:
24 //ctx.moveTo(width / 2, height / 2)
25 ctx.beginPath()
26 ctx.arc(width / 2, height / 2, 0.5 * height / 2, Math.PI, 2 * Math.
       PI, true);
27 ctx.stroke();
28
29 // Drawing eyes:
30 // Left:
31 ctx.beginPath()
32 ctx.arc(0.4 * width, 0.4 * height, 0.05 * height / 2, 0, 2 * Math.PI
       , true);
33 ctx.stroke();
34 // Right:
35 ctx.beginPath()
36 ctx.arc(0.6 * width, 0.4 * height, 0.05 * height / 2, 0, 2 * Math.PI
       , true);
37 ctx.stroke();
38
39
40   </script>
41 </head>
```

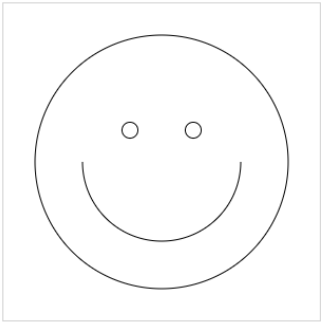

Figure 40: The resulting canvas smiley

16 Exam H2016

16.1 Section A: Multiple Choice Questions

1. *What is the correct HTML for making a text input field?*

```
<input type="text">
```

2. *How do you display hyperlinks without an underline?*
   Using CSS:

```
a {
    text-decoration: none;
}
```

3. *How do you display a border like this:*
   Top = 10 pixels
   Bottom = 5 pixels
   Left = 20 pixels
   Right = 1 pixel[1]

   The border are chosen in the order `top -> right -> bottom -> left`.

```
\\Answer to the question:
div {
    border-width: 10px 1px 5px 20px;
    border-width: thin medium thick 10px; // top: thin, right:
        medium, bottom: thick, left: 10px
    border-width: thin medium thick; // top: thin, right and left:
        medium, bottom: thick
    border-width: thin medium; // top and bottom: thin, right and
        left: medium
}
```

4. *What is the correct JavaScript syntax to change the content of the HTML element below?*

```
<p id="demo">This is a demonstration.</p>

// JavaScript:
document.getElementById("demo").innerHTML = "Hello World!";
```

5. *If you wanted to round the corners of a block element, which style property would you apply?*

   `border-radius:`

6. *Consider the table in Figure 41. Which line of code would create the last row of the table?*

[1] border-width: `https://www.w3schools.com/cssref/pr_border-width.asp`

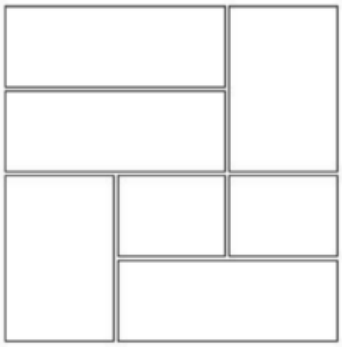

Figure 41: Table for Q6

```
<tr>
    <td colspan="2"></td>
</tr>
```

7. *How many alert boxes will the following loop produce?*

```
for (var index = 0; index <= 2; index++) {
    alert(index);
} // Will produce 3 alert boxes
```

8. *Which of the following is the code for accessing the contents of an input box named* ***`userData`*** *on a form?*

`document.forms[0].userData.value`

9. *Given a file structure as in Figure 42, if you are working on* ***`index.html`****, which of the following is the correct relative URL for* ***`lecture1.html`****?*

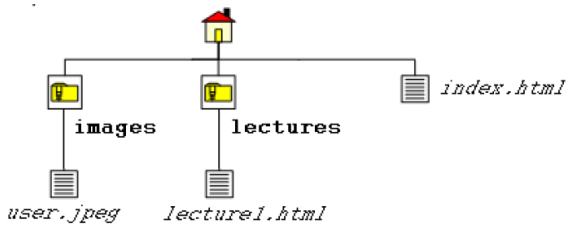

Figure 42: File structure for Q9

The correct is
`lectures/lecture1.html`
If specified with `../folder1/file1`, HTML will look in the folder above the current folder. Can be used multiple times.
Using `./` means that `"."` points to the same directory, and the slash `"/"` gives access to it.

```
// If you are on directory A, and want access to directory X:
- root
    |- a
        |- A
```

```
    |- b
    |- x
       |- X

// Absolute path:
"/x/X/picture.png"

// Relative path:
"./../x/X/picture.png"
```

10. *The syntax for adding a textfield of 3 rows, each of width 40 is:*

```
<textarea rows="3" cols="40">Some text</textarea>
```

11. *Indicate whether or not the following is true or false:*

    (A) This is a valid JSON:

```
{
    "id": 1,
    "name": "A green door",
    "price": 12.50,
    "tags": ["home", "green"]
} // This is valid!
```

    (B) *In a HTML form, the method attribute is used to specify the script that process the form data*
    False. It specifies **how** to send form-data. The form-data is sent to the page specified in the `action` attribute.

    (C) *Domain Name Servers (DNS) map symbolic computer names to their host names*
    False. They are mapped to an IP address

    (D) *XML elements cannot be empty*
    False.

12. *The output when running the following program*

```
var i = 25;
var j = "25";
var k = 2 + "5";        // gets converted to string: "25"
var l = "2" + "5";      // "25"

document.write(i==j);       // true
document.write(i===j);       // false. === means equal in value and type.
document.write(j==k);       // true
document.write(j===k);       // true
document.write(i==k);       // false
document.write(i===k);       // true
document.write(k==l);       // true
document.write(k===l);       // true
```

16.2 Section B: Longer Answer Questions

1. *Fill in the gaps:*

   A. *The purpose of the **transport layer** is to ensure the integrity of network communication*

   B. ***Dynamic routing** is a technique to selet the best route according to the network conditions at the time of transmission.*

   C. *What is the required attribute of the `<video>` element in HTML5, when the video is in a single format? `src, width, height` (Solution says only `src`)*

   D. *Consider the following JavaScript statement*
   `var myLinks = getElementsByTagName('a';`
   *What is the data type of the result? **An array***

   E. *If you want to control which element sits on the top (when elements overlap) you use the **z-index** property.*

   F. *The **content** part of the box (box model) is where text and images appear.*

2. *Select all valid XML tag names*:

```
1 <_Element1>       // not valid
2 <1Tag>            // not valid
3 <My.Element>      // valid
4 <#Elem&ent>       // not valid
5 <My-Element_Name>         // not valid
6 <XmL>             // not valid
```

3. *Fill in the table*

Image type	File extension	Compression (lossy/lossless)	Transparency	Animation	Colors (hundreds/millions)
GIF	`.gif`	Lossless	Yes	Yes	Hundreds
JPEG	`.jpg` or `.jpeg`	Lossy	No	No	Millions
PNG	`.png`	Lossless	Yes	No	Millions

Table 1: Table for Q3

4. *Write the appropriate position property value:*
   Static: This is the default setting - no special positioning.
   Absolute: The box is taken out of normal flow and no longer affects the position of the other elements. **Relative:** This property moves an element in relation to where it would have been in normal flow.
   Fixed: Configures the location of an element within the browser viewpoint - i.e. the element doesn't change position if scrolling content.

5. *Explain the differences between Canvas and SVG.*

   - In Canvas, elements are drawn programmatically, while in SVG it is part of the page's DOM
   - In Canvas, drawing is done with pixels, while in SVG it is done with vectors.
   - In Canvas, animations are not built-in, while in SVG animations are built-in.
   - Canvas has high performance for pixel-based drawing operations, while SVG is based on standard XML syntax, which provides better accessibility.

6. *Consider the following HTML file:*

```
<!DOCTYPE html>
<html>
<head>
<style>
/* unvisited link */

-----------------------------------------

[          ]{
   background-color: lightblue;
}

/* visited link */
[          ]{
   background-color: lightblue;
}

/* mouse over link */
[          ]{
   background-color: yellow;
}

/* selected link */
[          ]{
   background-color: yellow;
}
</style>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>
<p><a href="http://www.ntnu.no">NTNU</a></p>
</body>
</html>
```

Figure 43: HTML file for Q6

A. *Build the HTML tree*

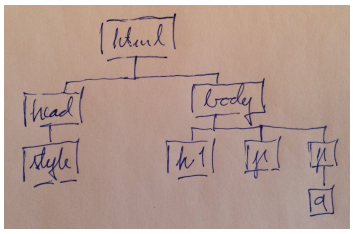

Figure 44: HTML tree

B. *Fill in the CSS pseudo-classes*

```
1 a:link { // Unvisited link
2     background-color: lightblue;
3 }
```

```
4
5  a:visited { // visited link
6      background-color: lightblue;
7  }
8
9  a:hover { // mouse over link
10     background-color: yellow;
11 }
12
13 a:active { // selected link
14     background-color: yellow;
15 }
16 // Note: Double colon, e.g. a::first is used for pseudo-elements
       . Represents elements in the HTML page that are not really a
       part of the rendered HTML. Ex.:
17 p::first-letter {
18     ...
19 }
20 p::before {
21     ...
22 }
```

16.3 Section C: Coding Related Questions

1. *Write code for a JavaScript function that:*

   a. *Reads in two numbers from an HTML form*

   b. *Determines which number is the largest*

   c. *Outputs to screen which number is the largest (in the format "44 is larger than 22"*

   d. *Write an appropriate HTML form which will call this function.*

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4          <title>Form</title>
5  </head>
6  <body>
7          <form action="forms/exForm.html" method="post" name="exForm"
               onclick="getNumbers()">
8          <!-- NB! Use name and not id when accessing from JavaScript
               in the way "document.formName.valueID.value" -->
9                  <input type="number" name="value1" min="1" max="10">
10                 <input type="number" name="value2" min="1" max="10">
11         </form>
12         <p id ="result"></p>
13         <script type="text/javascript" src="sectionC_1.js"></script>
14 </body>
15 </html>
16
17
18 // JavaScript:
19 function getNumbers() {
20         var num1 = document.exForm.value1.value
```

```
21          var num2 = document.exForm.value2.value
22
23          var result = document.getElementById('result')
24
25          if (num1 < num2) {
26                  //console.log(num2, " is larger than ", num1)
27                  result.innerHTML = num2 + " is larger than " + num1
28          }
29          else if (num1 == num2) {
30                  result.innerHTML = "The numbers are equal"
31          }
32          else {
33                  result.innerHTML = num1 + " is larger than " + num2
34          }
35  }
```

2. *Write a CSS rule that configures a class called "company" in bold, serif font, and 1.25em in size.*

```
1  .company { /* .company is the class selector. #elementID is the ID
       selector*/
2          font-family: serif;
3          font-weight: bold;
4          font-size: 1.25em;
5  }
```

3. *Consider the following HTML:*

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4          <title>Question 3, section C</title>
5  </head>
6  <body>
7          <p>In my younger and more vulnerable years my father gave me
               some advice that I've been turning over in my mind ever
               since. 'Whenever you feel like criticizing anyone,' he
              told me, 'just remember that all the people in this
              world haven't had the advantages that you've had.'</p>
8  </body>
9  </html>
```

Write the CSS pseudo-classes needed to:

a. *Set the text color to red, for the first line of the ¡p¿ element.*

b. *Set text color to blue, and the text-size to "xx-large" for the first letter of the <p> element*

c. *Insert the image "smiley.gif" before, and after, <p> elements.*

```
1  <head>
2          <title>Question 3, section C</title>
3          <style type="text/css">
4                  p::first-letter {
5                          color: blue;
6                          font-size: xx-large;
```

```
7                       }
8                       p::first-line {
9                               color: red;
10                      }
11
12
13                      p::before {
14                              content: url(smiley.gif);
15                      }
16                      p::after {
17                              content: url(smiley.gif);
18                      }
19              </style>
20      </head>
```

4. *Write the JavaScript that turns on and off the image of a lightbulb when clicked.*

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4           <title>Section C, question 4</title>
5   </head>
6   <body>
7           <h1>JavaScript can change images!</h1>
8           <img id="myImage" src="pic_bulboff.gif" onclick="changeImage
                ()" width="100" height="180">
9
10          <p>Click the light bulb to turn on/off the light</p>
11
12
13          <script type="text/javascript">
14                  function changeImage() {
15                          var image = document.getElementById('myImage
                                ')
16
17                          if (image.src.match("pic_bulboff.gif")){
18                                  image.src="pic_bulbon.gif"
19                          }
20                          else {
21                                  image.src = "pic_bulboff.gif"
22                          }
23                  }
24          </script>
25  </body>
26  </html>
```

5. *Fill in the missing JavaScript to produce the following:*

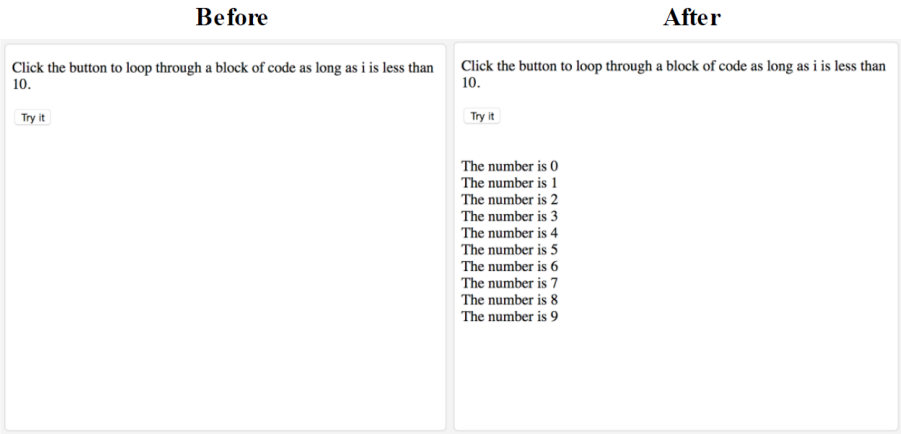

Figure 45: The result from the JavaScript code

```
<!DOCTYPE html>
<html>
<head>
        <title>Section C, Q5</title>
</head>
<body>
        <p>Click the button to loop through a block of code as long
            as i is less than 10.</p>

        <button onclick="myFunction()">Try it</button>

        <p id="demo"></p>

        <script type="text/javascript">
                function myFunction() {
                        var text = '';
                        var i = 0;
                        while(i < 10) {
                                text += '<br>The number is ' + i;
                                i++;
                        }
                        p = document.getElementById('demo');
                        p.innerHTML = text;
                }
        </script>
</body>
</html>
```

6. *Write JavaScript code to draw the following figure inside a* `<canvas>` *element*

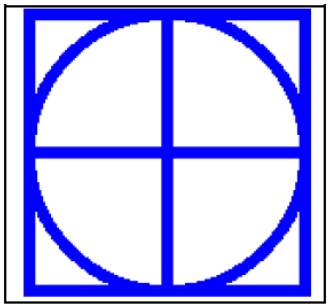

Figure 46: What you should draw with Canvas

```
<!DOCTYPE html>
<html>
<head>
        <title>Section C, Q6</title>
</head>
<body>
        <canvas id="canvas1" width="150" height="150" style="border:
            1px solid #d3d3d3">
                Your browser does not support canvas. <!-- This is
                    the fallback text-->
        </canvas>

        <script type="text/javascript">
                const canvas = document.getElementById('canvas1');
        const ctx = canvas.getContext('2d');

        // shorter syntax:
        var width = ctx.canvas.width;
        var height = ctx.canvas.height;

        // Drawing the borders:
        ctx.lineWidth = 5;
        ctx.strokeStyle="rgba(0,0,255, 1)";
        ctx.strokeRect(0, 0, width, height);

        // Drawing the circle:
        ctx.beginPath();
        ctx.arc(
                width / 2,      // x coordinate of center
                height / 2, // y coordinate of center
                height / 2, // radius
                0,      // starting angle
                2*Math.PI,      // ending angle
                true);  // true = anti-clockwise
        ctx.stroke();

                // Drawing the cross:
                ctx.beginPath();
                ctx.moveTo(0, height/2);
                ctx.lineTo(width, height/2); // horizontal line
                ctx.moveTo(width/2, 0);
```

```
40                  ctx.lineTo(width/2, height); // vertical line
41                  ctx.stroke()
42
43          </script>
44
45  </body>
46  </html>
```

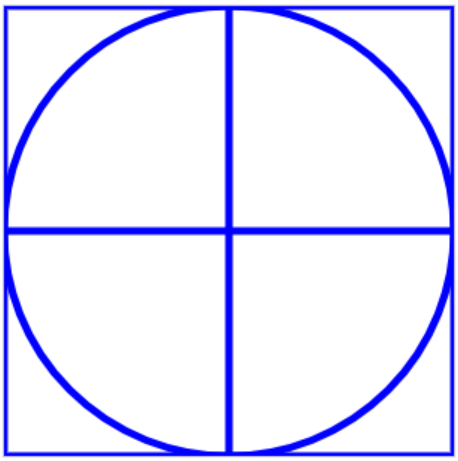

Figure 47: The resulting drawn canvas