# HTML, CSS & JavaScript

CHALMERS UNIVERSITY OF TECHNOLOGY

GÖTEBORGS UNIVERSITET

***DAT076/DIT126*** Web Applications

Adam Waldenberg

## *HyperText Markup Language (HTML)*

**A language for writing documents that contain links to other documents**

- A HTML document consists of a tree structure of elements – often referred to as the document tree. Elements can be ancestors, descendants, parents, children and siblings in relation to each other
- Based on the Standard Generalized Markup Language (SGML)
- Different variations and versions exist, but the one you should use today is HTML5
- HTML 5.2 recommendation ***(www.w3.org/TR/html52)***

## Early HTML time-line

### Many versions and variations

- **1991** Tim Berners-Lee invents HTML
- **1993** Dave Raggett drafts HTML+
- **1994** Formation of the HTML Working Group and W3C (World Wide Web Consortium)

## After W3C and HTMLWG

- **1995** HTMLWG defines HTML 2.0 ***(tools.ietf.org/html/rfc1866)***
- **1997** W3C HTML 3.2 ***(www.w3.org/TR/2018/SPSD-html32-20180315)***
- **1999** W3C HTML 4.01 ***(www.w3.org/TR/html401)***
- **2000** W3C XHTML 1.0 ***(https://www.w3.org/TR/xhtml1/)***
- **2004** Formation of WHATWG (Web Hypertext Application Technology Working Group)

## HTML5 versions

- **2008** WHATWG HTML5 First Public Draft
- **2014** W3C HTML5
- **2016** W3C HTML 5.1
- **2017** W3C HTML5.1 2nd Edition
- **2017** W3C HTML5.2 ***(www.w3.org/TR/html52)***
- **2018** W3C HTML5.3 Working Draft

## HTML elements

- Formed by an ***opening tag*** and an optional ***closing tag***
- Each element formed with a ***closing tag*** also has an optional ***content portion***
- The ***content portion*** can hold an arbitrary number of ***child-elements***
- Each element can have an arbitrary number of ***attributes***

***Examples:***

```
<b>Text to be put in bold</b>
<br>, <img src="path">
```

## *HTML attributes*

**Attributes are optional and modify the resulting behavior or appearance of the element or the resulting document data**

- Different elements accept different attributes
- Attributes are only valid in the ***opening tag***
- HTML5 adds the ability to define custom attributes by applying the **`data-`** prefix  to the attribute name

## *Extensible HyperText Markup Language (XHTML)*

**In enterprise development and JSF in particular we use XHTML instead of ordinary HTML**

- Based on XML, in essence resulting in a strict version of HTML
- Support for name spaces, including custom name spaces
- Case sensitive
- Attributes must be within quotes
- Tags can’t be left unclosed

## Document structure

```
<!DOCTYPE html>
<HTML>
       <HEAD>
       ...
       </HEAD>
       <BODY>
       ...
       </BODY>
</HTML>
```

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<html>
       <head>
       ...
       </head>
       <body>
       ...
       </body>
</html>
```

**What’s the difference?**

## Document head elements

**These are some of the more important elements used in the head section of HTML documents**

- Page title definition:

```
<title>Chalmers University of Technology</title>
```

- Meta tags that can give hints about properties of the document:

```
<meta charset="UTF-8">
<meta name="phrase" content="Hellow World!">
```

- Elements to include page resources:

```
<link rel="stylesheet" href="main.css">
<script type="text/javascript" src="file.js"></script>
```

## Document body elements

**These are some of the more important elements used in the body section of HTML documents**

- Paragraphs:

```
<p>Hello World!</p>
```

- Headers:

```
<h1>Important</h1> <h2>Less important</h2> … <h6>...</h6>
```

- Styling:

```
<em>Emphasized text</em>
<strong>Bold text</strong>
```

## More document body elements

- Unordered bullet point list:
  `<ul><li>First</li><li>Second</li></ul>`
- Ordered (numbered) list:
  `<ol><li>First</li><li>Second</li></ol>`
- Images and video:
  `<img>`, `<video>`
- Plus many, many more - in total there are approximately 110 different definable elements in HTML5

## *Dividers and semantic tags*

- A ***`<div>`*** tag divides the page up into sections
- For specialized sections there are semantic tags: ***<header>, <nav>, <section>, <article>*** to name a few
- Sometimes you will see ***<span>*** used as a section divider. This is not the correct use.

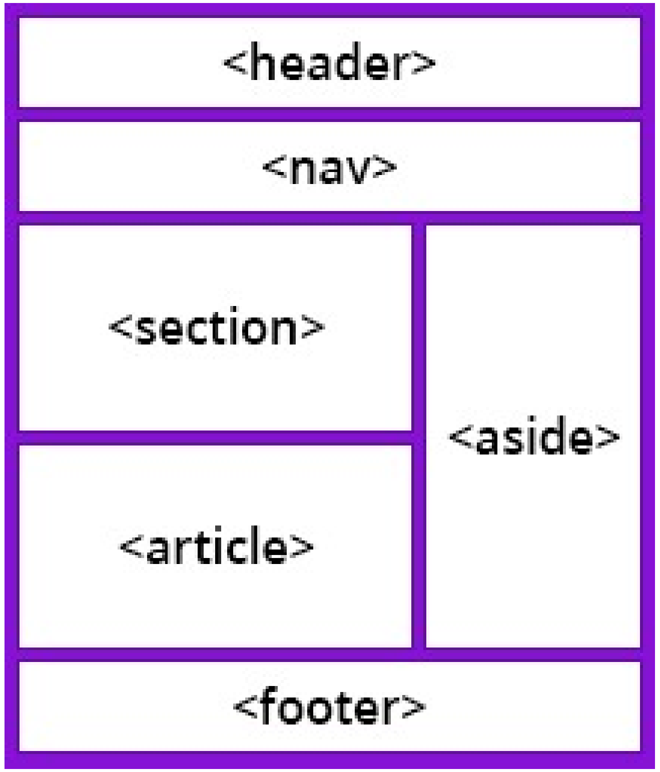


*Image from www.bitdegree.org*

## The anchor tag

```
<a href =”https://www.example.com”>Example.com</a>
```

- Creates a clickable link that will make the browser fetch and render the document in the specified target
- Can also be a placeholder for a link. However, so can any HTML element when combined with JavaScript and an onclick event.
- Can define anchor points with the ***name*** attribute

## Exploring the anchor tag

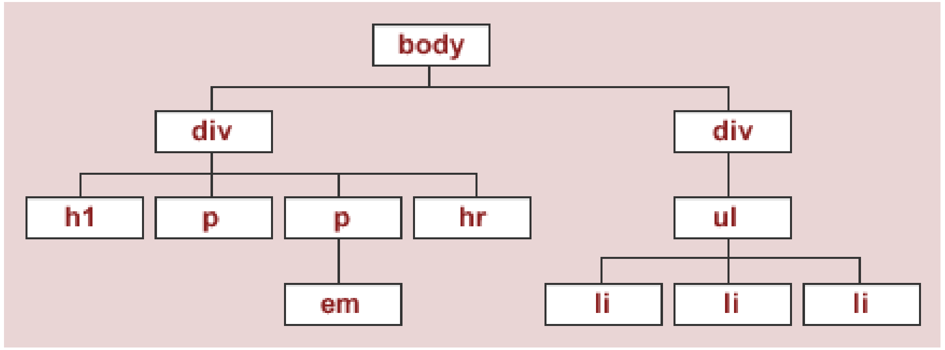
- Can hold an absolute URL:
  ***`https://www.chalmers.se/en/about-chalmers/Pages/default.aspx`***
  ***`/en/about-chalmers/Pages/default.aspx`***

- Can hold a relative URL:
  ***`default.aspx`***
  ***`../default.aspx`***

- Never include the hostname or IP address for local links within your own site in your ***`href`*** attributes. You should only specify it when you need to link to external sites

# A HTML body example

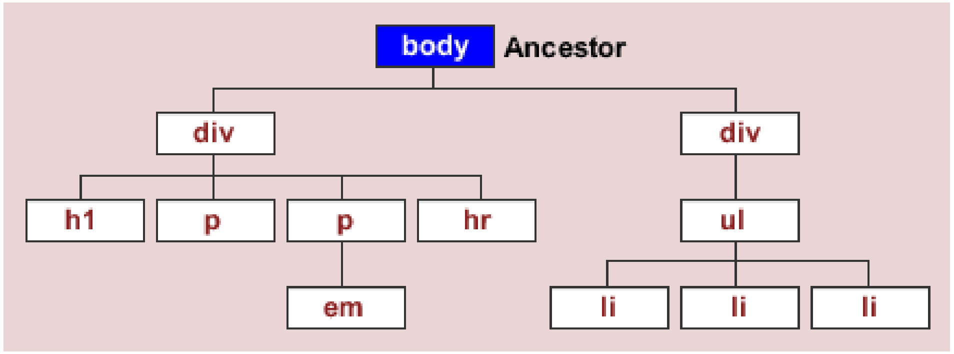
```
<body>
    <div id="content">
        <h1>Headinghere</h1>
        <p>Hello Wold!</p>
        <p>Lorem <em>ipsum</em></p>
    </div>
    <div id="nav">
        <ul>
            <li>item 1</li>
            <li>item 2</li>
            <li>item 3</li>
        </ul>
    </div>
</body>
```



*This is the layout of the document tree of this particular HTML document*

# A HTML body example : Ancestors

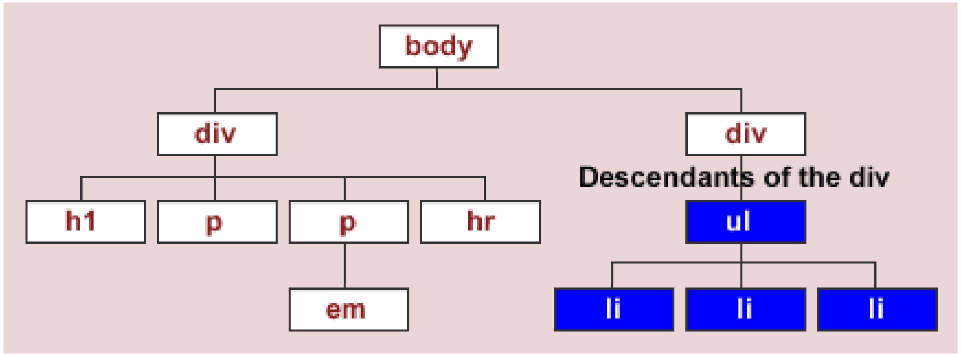
```
<body>
    <div id="content">
        <h1>Headinghere</h1>
        <p>Hello Wold!</p>
        <p>Lorem <em>ipsum</em></p>
    </div>
    <div id="nav">
        <ul>
            <li>item 1</li>
            <li>item 2</li>
            <li>item 3</li>
        </ul>
    </div>
</body>
```



*An **ancestor** refers to any element that is connected, no matter how many levels higher up in the document tree*

# A HTML body example : Descendants

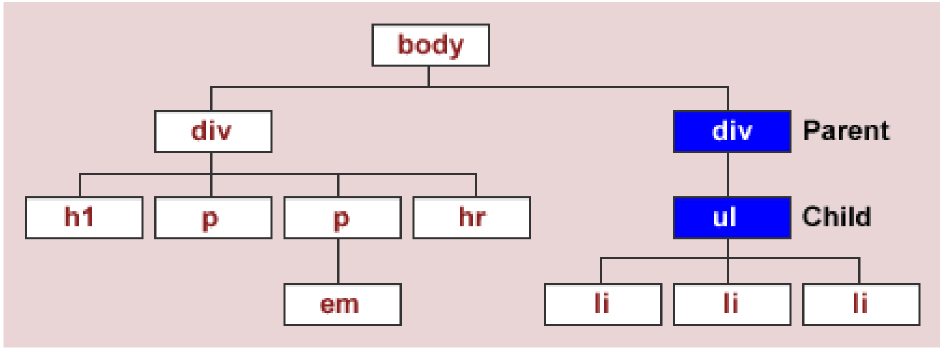
```
<body>
    <div id="content">
        <h1>Headinghere</h1>
        <p>Hello Wold!</p>
        <p>Lorem <em>ipsum</em></p>
    </div>
    <div id="nav">
        <ul>
            <li>item 1</li>
            <li>item 2</li>
            <li>item 3</li>
        </ul>
    </div>
</body>
```



*A **descendant** refers to any element that is connected but lower down the document tree - no matter how many levels lower*

## A HTML body example : Parents & Children

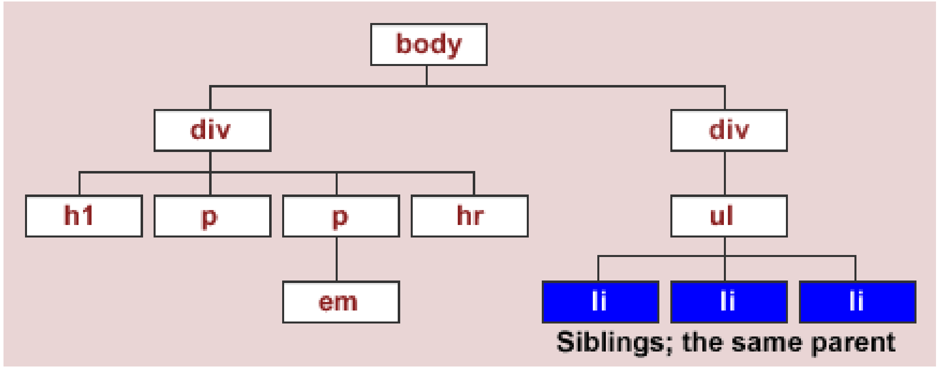
```
<body>
    <div id="content">
        <h1>Headinghere</h1>
        <p>Hello Wold!</p>
        <p>Lorem <em>ipsum</em></p>
    </div>
    <div id="nav">
        <ul>
            <li>item 1</li>
            <li>item 2</li>
            <li>item 3</li>
        </ul>
    </div>
</body>
```



*Elements that are directly related to each other on the previous and next level*

## A HTML body example : Siblings

```
<body>
    <div id="content">
        <h1>Headinghere</h1>
        <p>Hello Wold!</p>
        <p>Lorem <em>ipsum</em></p>
    </div>
    <div id="nav">
        <ul>
            <li>item 1</li>
            <li>item 2</li>
            <li>item 3</li>
        </ul>
    </div>
</body>
```



- *A **sibling** is an element that shares the same parent with another element*

## *Is there a good reference page?*

**Yes there is! WHATWG has a *living standard* document that they update continuously**

- If you need to look something up, you can visit: ***html.spec.whatwg.org***

- There is also a PDF version available for download: ***html.spec.whatwg.org/print.pdf***

## Let’s do some examples!

***jsitor.com***

## Cascading Style Sheets (CSS)

- A series of rules to modify the appearance of the generated document – later rules override earlier rules, unless they are a less exact match.
- Rules are also merged with each other if the properties inside do not collide
- Defined and released by W3C in 1996
- Prior practice to CSS was to design sites by relying heavily on tables and the now deprecated ***background***, ***bgcolor*** and ***bordercolor*** attributes
- Browsers managed to achieve proper compatibility around the shift of the millennium

## CSS syntax

### The syntax of a CSS rule looks like this

```
selector {
         property1: value1 value2 … value3;
         property2: value;
         …
         property3: value;
}
```

- A selector controls which element(s) the rule will be applied to. Selectors can be combined using combinators, examples include **`>, ~, +`**
- Selector can consist of an element type with an optional class, id or attribute
- Selectors can also consist of only a class, id or attribute definition

## CSS rules explained

**CSS rules, properties and the selector syntax can be hard to explain – the best way is to learn by example**

**So lets do a few, shall we!**

## CSS rules explained continued

```
p {
        font-size: 20pt;
        …
}
```

**Changes the font size in all paragraph texts in the whole document**

## CSS rules explained continued

```
div.box {
        background-color: gray;
        …
}
```

**Changes the background color of all div’s in the document that have the class `box` applied to them**

## CSS rules explained continued

```
div > h1 {
        color: gray;
        …
}
```

**Changes the text color of all h1 headers in the document that are direct children of a div**

**The child combinator `>` selects a child element**

## CSS rules explained continued

```
article, div > h2, p {
        background-color: green;
        …
}
```

**Changes the background color to green on all articles, all h2 headers that are children of a div and all paragraphs**

**The combinator `,` allows to select multiple CSS rules for the same set of CSS properties**

## CSS rules explained continued

```
p:last-child {
        font-weight : 800;
        …
}
```

**Changes the font weight of all paragraphs that are the last child of their parent**

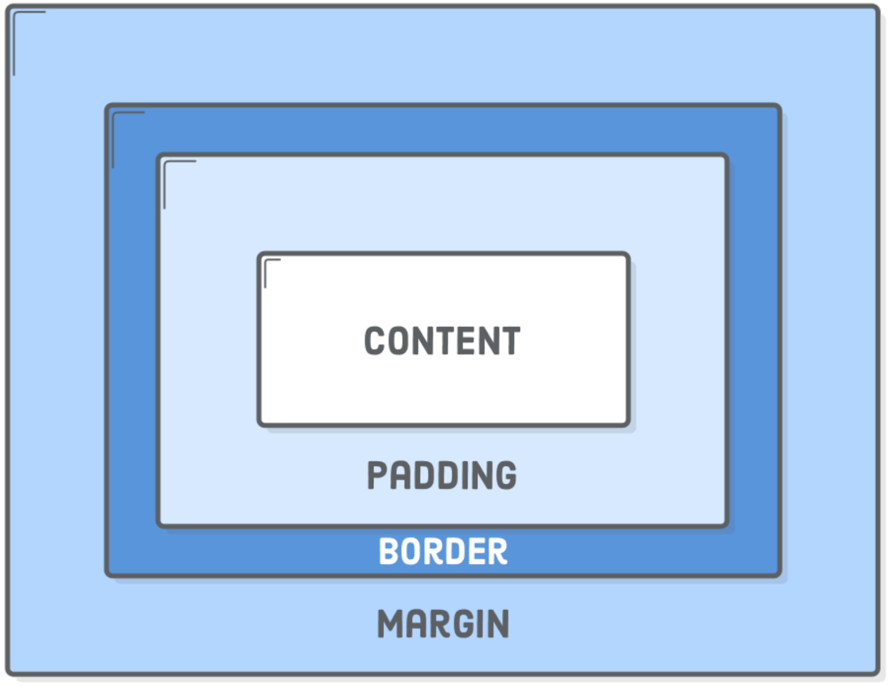
**last-child is refered to as a CSS psuedo class and there are a number of these available for use that select elements based on different criteria**

## CSS selector syntax reference

**There are so many ways to write the selectors it’s impossible to cover all of it**

- For a somewhat exhaustive list, please refer to: ***www.w3.org/TR/selectors-3/#selectors***

## CSS margins and padding



- CSS rules can pad and put margins around elements in the document via the following properties:

  ***padding***
  ***padding-[top/left/righ/bottom]***
  ***margin***
  ***margin:-[top/left/right/bottom]***

- The values given to these properties can be specified in different units:

  ***%, px, pt, em***

## CSS Media queries

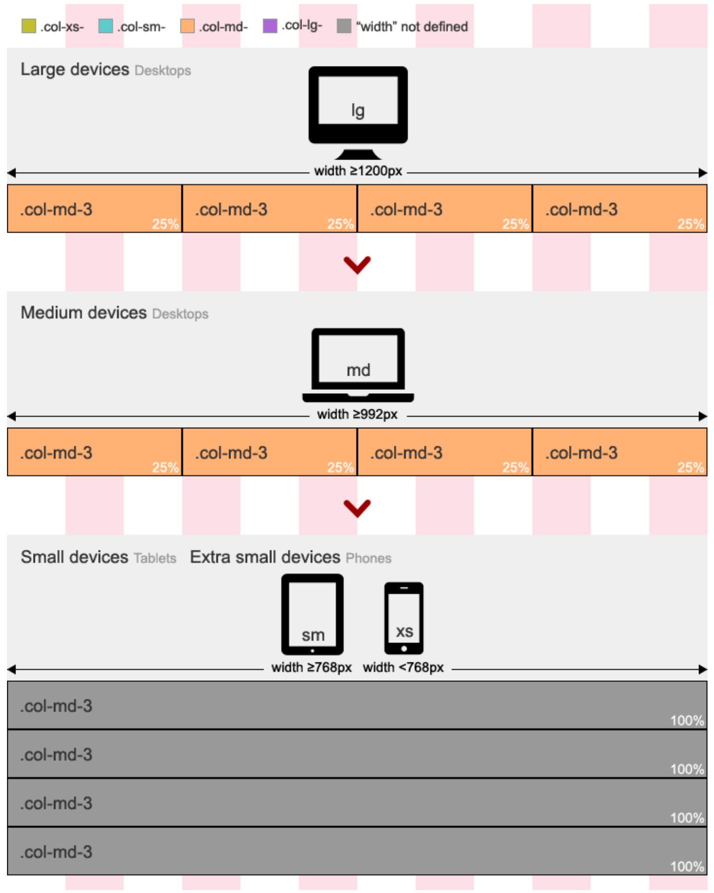
```
@media all and (min-width: 601px) {
     div.example {
          font-size: 80px;
     }
}

@media all and (max-width: 600px) {
     div.example {
          font-size: 30px;
     }
}
```

- What is happening here?
- Media queries are used for responsive web design
- Generally  very hard to get right and very error prone
- The solution is to use a CSS framework such as Bootstrap which solves this for us and offers a responsive grid system that we can use

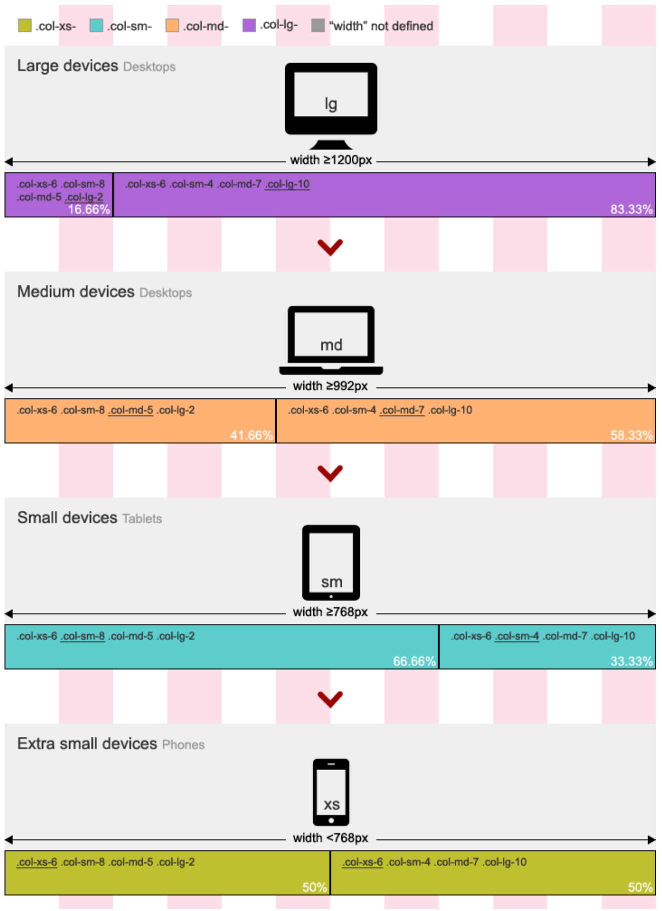
# The Bootstrap grid system

- The Bootstrap grid system is controlled by applying classes to elements
- There are classes for extra small, small, medium and large devices
- There are “12 units” in width in the Bootstrap grid system
- The only way to really understand it is to play with it and use it. In Assignment 2 you get a chance to get acquainted with the grid system



# The Bootstrap grid system continued

- A more sophisticated example
- Each device type has a different unit width defined, resulting in slightly different layouts on the different devices or resolutions
- PrimeFaces which is a component framework  you are recommended to use in your project if you choose to use JSF has a grid system that works similarly


.col-xs-
.col-sm-
.col-md-
.col-lg-
"width" not defined
Large devices Desktops
lg
width ≥1200px
.col-xs-6 .col-sm-8 .col-md-5 .col-lg-2
16.66%
.col-xs-6 .col-sm-4 .col-md-7 .col-lg-10
83.33%
Medium devices Desktops
md
width ≥992px
.col-xs-6 .col-sm-8 .col-md-5 .col-lg-2
41.66%
.col-xs-6 .col-sm-4 .col-md-7 .col-lg-10
58.33%
Small devices Tablets
sm
width ≥768px
.col-xs-6 .col-sm-8 .col-md-5 .col-lg-2
66.66%
.col-xs-6 .col-sm-4 .col-md-7 .col-lg-10
33.33%
Extra small devices Phones
xs
width <768px
.col-xs-6 .col-sm-8 .col-md-5 .col-lg-2
50%
.col-xs-6 .col-sm-4 .col-md-7 .col-lg-10
50%

## JavaScript

- **Not** related to Java (the creator was a naughty boy)
- First version written in 1995 and created in 10 days
- Originally completely interpreted, but modern variants use JIT compilers
- Single-threaded, but  can be threaded under Node.js
- Dynamically typed
- Completely open and unprotected, we can monkey-patch and modify everything – even external dependencies

## JavaScript continued

### Some example code

```
function myFunc(theObject) {
   theObject.brand = "Toyota";
}

var mycar = {
   brand: "Honda",
   model: "Accord",
   year: 1998
};

console.log(mycar.brand);
myFunc(mycar);
console.log(mycar.brand);
```

# JavaScript continued

## Dynamic typing and safety

```
var dog = { name : “Spot”, breed : “Dalmatian” };
```

- This is called a dictionary and is a key/value store
- No set properties are fixed. JavaScript allows us to change them and to set them to any type. If we set a non-existent property, it gets created

```
dog.name = -3;
dog.age = 17;
dog.age = “Seventeen”;
delete dog.breed;
```

- All of the above operations would be considered valid

# JavaScript continued

## So how similar is it to Java?

- Similar syntax resembling the C/C++/Java code style
- ECMAScript 6 and upwards brings many new features like template strings, imports, classes, yields and promises
- We will cover some of these features in the React lecture later in the course
- Unfortunately does not support any proper object orientation, however you can overwrite parts of a class with the usual monkey patching that you have always been able to do in JavaScript

## JavaScript continued

- Has the usual arithmetic operators (`+, -, *, /, %, ++, --`)
- Same logical operators (`!, &&, ||`)
- JavaScript offers a decent string library with string operations:

```
var str = “The quick brown fox”
str.slice(4, 19) – returns “quick brown fox”
str.split(“ “) - returns [“The”, “quick”, “brown”, “fox”]
```

- As with everything else, you can mix any type you want in arrays
- Watch out for the differences between double-equals and triple-equals

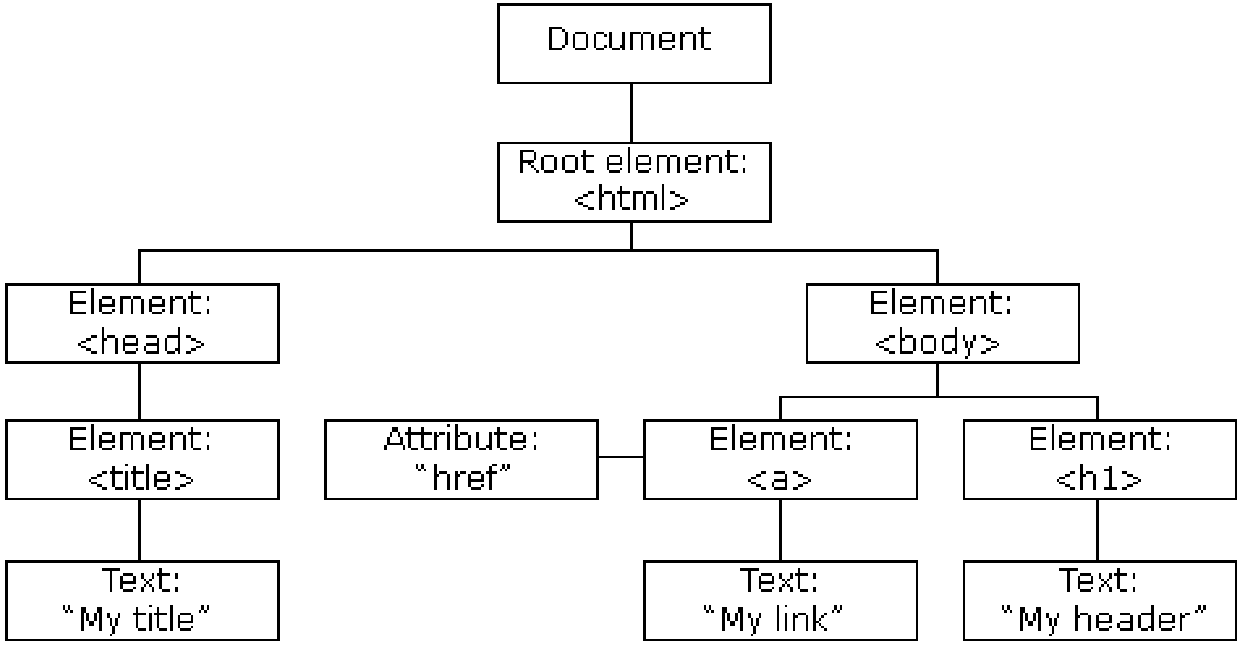
## *JavaScript continued*

**Lets take a look at the behavior of double equals vs triple equals**

***jsitor.com***

## The Domain Object Model (DOM)

- The internal representation of the HTML document inside the browser and a JavaScript object



## *Manipulating and querying the DOM*

**We use JQuery to manipulate and query the DOM. Why? Because it is a lot more streamlined to use compared to the built in API**

- JQuery simplifies everything to such an extent that it has basically become a standard. It is used by almost every JavaScript framework on the web. This also includes most component frameworks for JSF and Java EE / Jakarta EE
- Cached with in browsers and included as a library by default
- Has an amazing documentation:
  ***api.jquery.com***

## Unobtrusive JavaScript

**You are not allowed to have obtrusive JavaScript code in your HTML documents (it's considered bad practice)**

- This means that other than having to include a JavaScript file in the document, there should be no JavaScript code visible inside the document

  JQuery makes it simple to achieve – just attach a callback to the ready event:

```
$(document).ready(function() { … }
```

- The ***`document.ready`*** callback is called by JQuery when the document has finished loading and the DOM is fully available – allowing you to populate it with any additional events or elements you might need

## Modifying a collection of elements

```
$(document).ready(function() {
    $("p").text(“this is a paragraph”);

    $("article > p").text(function(index) {
        return "number " + (index + 1);
    });
});
```

- *The selector is similar to a CSS selector and returns a collection of matching elements*

- *In this example we first find all paragraph elements in the HTML document and modify their text to “this is a paragraph”.*

- *Next, we look up all paragraphs  that have a article as parent and change their text to “number <index>” where index is the index of the returned collection*

## *JavaScript events*

```
$(document).ready(function() {
   $("button").click(function() {
      console.log(“button was pressed”);
   });
});
```

- *In this example we attach an **onclick*** event to all button elements in the document. When the user clicks on the  button, the message ***“button was pressed”*** will be printed on the JavaScript console of the browser

## CSS3 Animations

**CSS allows you to animate and transform the state of elements**

```
p {
    transition: transform 3s;
}

p:hover {
    transform: rotate(45deg);
}
```

- *This example initiates a transition (animation) and rotates all paragraph 45 degrees whenever a user hovers the mouse pointer over them*

## CSS3 Animations

**CSS3 also supports the animation property in conjunction with keyframes to allow you to define complex animations**

```
div {
    animation-name: example;
    animation-duration: 10s;
}

@keyframes example {
  from {background-color: red;}
  50% {background-color: black;}
  to {background-color: yellow;}
}
```

- *This example instantly starts animations on page load – transitioning the background color of **all the divs** on the page from red to black to yellow under a period of 10 seconds*

## *CSS3 Animation – an extreme example*

**The only limit to what you can do is your imagination – with some tweaks and work you can do almost anything**

- Not only do CSS animations support simple transitions – they can even rotate elements in 3D and set a perspective
- This first-person shooter demo shows what you can actually do: ***keithclark.co.uk/labs/css-fps***