Web Dev

## Introduction

This document covers

Introduction

## Basics

### DOM

The DOM is a tree of objects that represent a web page. The objects the DOM can be accessed from and manipulated by JavaScript. While the traditional html, head and body tags are optional in HTML5 browsers will add them if you leave them out. The following shows a simple HTML5 document

<!DOCTYPE html>

<!-- The html element is optional in html5-->

<html lang=en>

    <head>

        <meta charset="utf-8">

        <title>Hello World</title>

        <!-- head hold information about the page-->

    </head>

    <body>

        <!-- The body holds the actual content-->

        <p>Terve Maailmalle</p>

    </body>

</html>

### CSS

CSS can be specified in three ways as this fragment shows

  <head>

        <meta charset="utf-8">

        <title>Hello World</title>

        <!-- 1. External Style Sheet-->

        <link rel="stylesheet" href="../mystyle.css">

        <!-- 2.  Embedded Style sheet -->

        <style>

            h1 { text-align: center; }

        </style>

    </head>

    <body>

        <!-- 3. Inline Style-->

        <h1>Suomi</h1>

        <p  style="color: blue;">Terve Maailmalle</p>

    </body>

### JavaScript

JavaScript can also be specified in three ways.

<html lang=en>

    <head>

        <meta charset="utf-8">

        <title>Hello World</title>

        <!-- 1. External Script-->

        <script src="../../js/examples.js"></script>

        <!-- 2.  Embedded Script -->

        <Script>

            addEventListener("load", function(ev) {

                // getElementById searches decendants

                console.log(document.getElementById("bd").nodeName);

            });

        </Script>

    </head>

    <body>

        <h1>Suomi</h1>

        <!-- 3. Inline -->

        <button

            onclick="document.getElementById('text').innerHTML = Date()" id="bd">

            Click Me</button>

        <p id="text"></p>

    </body>

</html>

### A decent Template

The following shows how we might put together a basic page that imports an external style sheet and an external java script file

<!DOCTYPE html>>

<html lang=en>

    <head>

        <meta charset="utf-8">

        <title>Hello World</title>

        <link rel="stylesheet" href="./style.css"/>

        <script src="./scripts.js"></script>

    </head>

    <body>

        <h1> Hello World</h1>

        <button onclick="external()">Click Me</button>

    </body>

</html>

## HTTP

When one types in a URL into a browser the browser first takes the domain name part and uses it to resolve an address using DNS. If this resolution is successful, the next step is to attempt to open a TCP connection on port 80 of the host. Port 80 is the default port for HTTP requests. Finally, the browser sends a GET request which might look something like the following.

GET /HelloWorld.html HTTP/1.1

Host: somehost.co.uk

User-Agent: Chrome

The server then sends a response back that looks something like the following.

The server response

HTTP/1.1 200 OK

Content-Length: 62314

Content-Type: text/html

<!doctype html>

…

Everything from <!doctype html> down is parsed as html and used to build the DOM that is rendered to the screen.

### Methods

The first part of the request defines the method. In the below request it is GET

**GET** /HelloWorld.html HTTP/1.1

The list of valid methods includes

* GET: get resource
* DELETE: delete resource
* PUT: replace resource
* POST: send information to resource

GET is used for requests that just request information. If any request makes changes or has side effects, it should use POST.

### Arguments

We can pass extra arguments with an HTTP request. If the method is GET, then the extra arguments are included in the URL itself.

GET / HelloWorld.html**?name=kenny&age=21** HTTP/1.1

Note how the query string arguments start with a question mark. Ampersands separate arguments. If the method is POST, then the arguments are put in the request not in the URL

POST/ HelloWorld.html HTTP/1.1

Content-Length: 24

Content-type: application/x-www-form-urlencoded

**name=kenny&age=21**

### Response Format

HTTP/1.1 \*200\* OK

The number I have highlighted in bold is the code. Codes starting with 2 are success codes and codes starting with 4 are error codes. 404 is the famous resource not found code.

### Get Requests

A single page may require tens of other resources from the same server to render (JavaScript, CSS, images etc). Each one will be retrieved by GET and the browser will make requests in parallel.

### XMLHttpRequest

Enables java script to make http requests to server. Does not need to be XML based despite the name. One creates an XMLHttpRequest object, open it and send it. Typically, we want to use the asynchronous version of send and add a listener on the requests load event so we don’t block the interface. We can use XML or JSON for the payload. JSON is easier to read and write by programmers and machines.

### Sandboxing

Browser prevents scripts making HTTP requests to other domains.

### Request Format

Rather than modelling remote calls as RPC if we model them as objects than encapsulate the request parameters we can more easily map them to http requests and can hence take advantage of things like client side caching.

### HTTP Details

## DOM

To a web browse each tag is a box. Boxes have content which can be other tags. There are two kinds of box - block boxes and inline boxes. Block level elements such as <div>,<p>,<table>,<list> have a line break before and after then. Inline elements such as <span>,<strong> can have multiple such element inline on a single row.

### Elements

#### Div

Div is a block level element meaning it has a line break before and after it. Such a block can represent navigation bar, banner or anything else that takes a chunk of the page

#### Span

A span is an inline element appearing as part of a paragraph

## CSS

The three basic selector types

    <style>

        /\* Class Selector \*/

        .redText {

            color: cornflowerblue;

        }

        /\* id selector \*/

        #headingThree {

            color: crimson;

        }

        /\* html selectors \*/

        p {

            color: green

        }

    </style>

## React

JSX is a syntax extension to JavaScript. JSX expressions are compiled into regular JavaScript and as such they can be returned from functions and assigned to variables etc. JSX compiles to calls to React.createElement. The result is an object. We can think of something like this.

const element = (

<h1 className="greeting">

Hello, world!

</h1>

);

Generates

const element = {

type: 'h1',

props: {

className: 'greeting',

children: 'Hello, world!'

}

};

### Components

Component names must always start with a capital letter and they must behave like pure functions in that they never try and modify their props

function Welcome(props) {

return <h1>Hello, {props.name}</h1>;

}

#### State and lifecycle

class Clock extends React.Component {

constructor(props) {

super(props);

this.state = {date: new Date()};

}

➋componentDidMount() {

this.timerID = setInterval(

() => this.tick(),

1000

);

}

➌componentWillUnmount() {

clearInterval(this.timerID);

}

tick() {

➊ this.setState({

date: new Date()

});

}

render() {

return (

<div>

<h1>Hello, world!</h1>

<h2>It is {this.state.date.toLocaleTimeString()}.</h2>

</div>

);

}

}

ReactDOM.render(

<Clock />,

document.getElementById('root')

);

➊ Never update state directly. Always use the set state method. This method merges values into the state.

➋ Invoked when the Clock is added to the DOM

➌ Invoked when the Clock is removed form the DOM

#### Forms

<https://reactjs.org/docs/forms.html>