

# Informationsteknologi B | Afsluttende projekt

## - Rapport -

---

### Tal Konvertering

#### Hexadecimale Talsystem

**Konverter**

Dit input:  
Dit resultat:

#### Binære Talsystem

**Konverter**

Dit input:  
Dit resultat:

### Hexadecimale Talsystem

**Konverter**

Dit input var: DAD  
Resultat: 3501

**Elever:** Ferhat Akdeniz & Nathaniel Finn Michel Risum

**Vejleder/lærer:** Karl G Bjarnason (kgb)

**Afleverings dato:** 10.05.2019

**Skole:** HTX - Roskilde

**Klasse:** 3.4 Mat/IT

**Fag:** Informationsteknologi B

# Indholdsfortegnelse

<b>1. Indledende aktivitet</b>	<b>4</b>
1.1. Introduktion	4
1.2. Ressourceplanlægning	5
1.3. Analyse	5
1.4. Problemformulering	6
<b>2. Teori</b>	<b>7</b>
2.1. Indledende afsnit	7
2.2. Specifikation af målgruppe	7
2.3. Konvertering	7
2.4. Valg af værktøjer	9
2.4.1. Programmeringssprog	9
2.5. HTML	10
2.6. CSS	10
2.7. Javascript	11
2.8. Udviklingsmiljø	11
2.8.1. Indledende afsnit	11
2.8.2. GitHub	11
2.8.3. Brackets	12
2.8.4. Atom	12
2.9. Laswells Kommunikationsmodel	13
2.10. User Story	13
2.10.1. User story 1	14
2.10.2. User story 2	14
2.10.3. User story 3	14
2.11. Usability	14
2.12. Extreme programming	15
2.13. OSI-modellen	16
<b>3. Iterationer af system udviklingsaktiviteter</b>	<b>17</b>
3.1. Indledende afsnit	17
3.2. Første iteration af systemudviklings-aktiviteterne	17
3.2.1. Planlægning	17
3.2.2. Design	17
3.2.3. Brugergrænseflade	17
3.2.4. Programmets virkemåde	18
3.2.5. Test	20
3.2.6. Resultatopgørelse	21
3.3. 2. Iterations af systemudvikling	21
3.3.1. Planlægning	21
3.3.2. Kontrol af krav og testproducerer	22

3.3.3.	Design	22
3.3.4.	Implementering	23
3.3.5.	Test	24
3.3.6.	Resultatopgørelse	25
3.4.	3. Iteration af systemudviklingsaktiviteterne/prototype	25
3.4.1.	Planlægning	25
3.4.2.	Kontrol af krav og testproducerer	26
3.4.3.	Design	26
3.4.4.	Implementering	26
3.4.5.	Test	33
<b>4.</b>	<b>Resultatopgørelse</b>	<b>34</b>
4.1.	Indledende afsnit	34
4.2.	Hvad er opnået?	34
4.2.1.	Problemformulering	34
<b>5.</b>	<b>Evaluerende tanker</b>	<b>34</b>
<b>6.</b>	<b>Konklusion</b>	<b>35</b>
<b>7.</b>	<b>Kildeliste</b>	<b>35</b>
<b>8.</b>	<b>Bilag</b>	<b>36</b>
8.1.	Tidsplanlægning	36
8.2.	Kildekode	38
8.2.1.	Første iteration	38
8.2.2.	Anden iteration	39
8.2.3.	Tredje iteration	40
8.3.	Licenser	51
8.3.1.	Brackets	51
8.3.2.	Atom	51
8.3.3.	Licens - produkt	52

# 1. Indledende aktivitet

## 1.1. Introduktion

Dette er et eksamensprojekt i faget informationsteknologi. Temaet i projektet er valgfrit, således at problemafgrænsningen og problemformuleringen selv skal opstilles. Dette projekt tager udgangspunkt i et produkt, som andre studerende kan lære noget af, et såkaldt læringsprodukt.

I denne rapport er temaet et læringsprodukt, hvor der arbejdes med konvertering mellem talsystemer, som skal vises på et website. Herunder er udgangspunktet en konvertering fra hexadecimal til decimal, hvorefter der tilføjes endnu en konvertering fra binære til decimal. Websitet skal fungerer, som et formidlingsprodukt sammen med rapporten.

Konvertering af tal fylder enormt meget i vores hverdag, men problemet er, at ikke alle er klar over hvad der står på eksempelvis en fejlkode på computerskærmen, men dog kan det somme tider give god mening at have en forståelse for hvad der gemmer sig bag disse talsystemer. Denne rapport vil belyse alle disse områder, ikke mindst vil der udvikles et program, som viser en konvertering mellem forskellige talsystemer samt et website.

I dette projekt tages der højde for de faglige mål, herunder følgende;

- redegøre for grundlæggende funktioner af it-komponenter (hardware og software) og samspillet mellem dem
- redegøre for samspillet mellem it-komponenter og bruger
- redegøre for samspillet mellem it-komponenter og de fysiske omgivelser
- beskrive sammensatte systemer opbygget af virtuelle niveauer
- analysere og beskrive sikkerhedsbehov og risikofaktorer ved brug af et givent it-system
- vælge og bruge it-komponenter som værktøj til løsning af et problem med relation til elevens, uddannelsens, virksomhedens og samfundets brug
- anvende it som interaktivt medie til dokumentation og kommunikation
- redegøre for innovative it-systemer sammenholdt med egne it-løsninger
- realisere prototyper på it-systemer, herunder kunne installere, konfigurere og tilpasse relevante it-komponenter.

For at opnå ovenstående mål benyttes en række forskellige metoder.

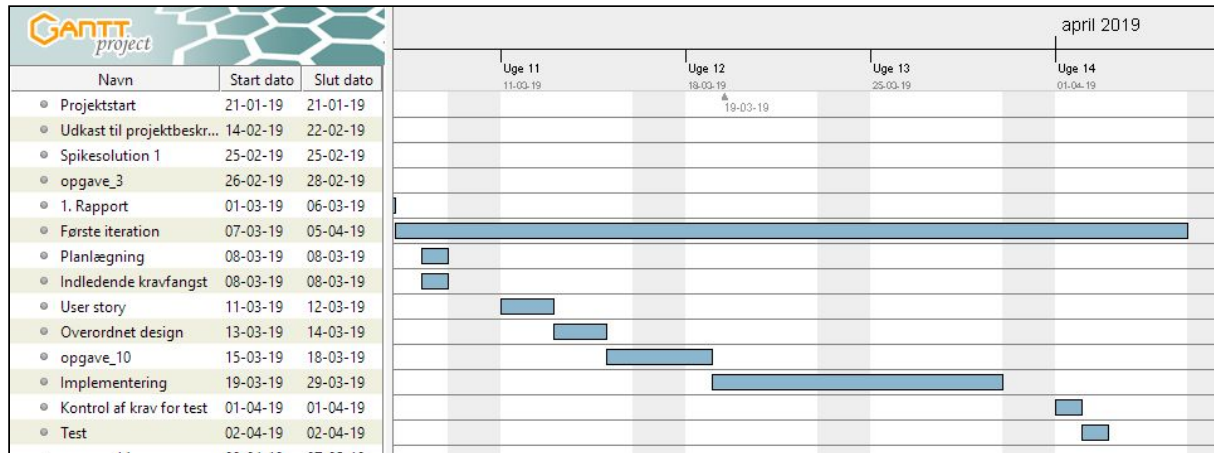
Blandt andet bruges en systemudviklingsmetode udviklet på HTX → Roskilde

Derudover benyttes der kommunikationsmodeller i forhold til målgruppen.

Gruppen består af to elever fra HTX - Roskilde, som har en studieretning, som hedder; Matematik A - Informationsteknologi B - Programmering C. Gruppen består af: Ferhat Akdeniz og Nathaniel Finn Michel Risum.

## 1.2. Ressourceplanlægning

Vi laver herunder en ressourceplanlægning, hvor vi medtager opgaver samt alle opgaver fra udviklingen af hele første iteration.

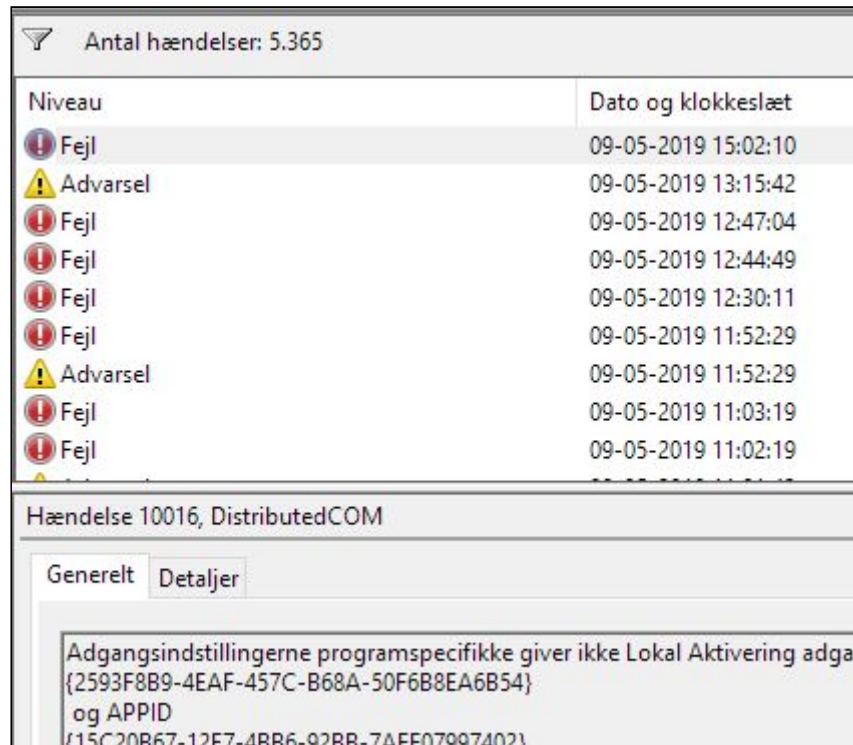


En udvidet tidsplan over projektet kan findes i bilag<sup>1</sup>.

## 1.3. Analyse

I mange digitale sammenhænge kan man af adskillige årsager støde på fejlkoder. Og det er uanset om man så støder på BSoD - Blue Screen of Death -, driver fejl eller noget helt tredje. Det er faktisk ganske normalt, og man skal faktisk ikke gå meget længere end Windows Logbog for f.eks. at støde på nedenstående:

<sup>1</sup> Tidsplanlægning - Punkt 7.1.



Niveau	Dato og klokkeslæt
Fejl	09-05-2019 15:02:10
Advarsel	09-05-2019 13:15:42
Fejl	09-05-2019 12:47:04
Fejl	09-05-2019 12:44:49
Fejl	09-05-2019 12:30:11
Fejl	09-05-2019 11:52:29
Advarsel	09-05-2019 11:52:29
Fejl	09-05-2019 11:03:19
Fejl	09-05-2019 11:02:19

Hændelse 10016, DistributedCOM	
Generelt	Detaljer
Adgangsindstillingerne programspecifikke giver ikke Lokal Aktivering adga {2593F8B9-4EAF-457C-B68A-50F6B8EA6B54} og APPID {15C20B67-12F7-4BB6-92BB-7AFF07997402}	

De fejlkoder der ses på billedet angives i det hexadecimale talsystem. Hvis man har lyst til at rette op på disse fejl, har man højst sandsynligt en interesse for IT-systemer og drift af disse. For overhovedet kan kunne slå en løsning op i et datablad skal man dog først og fremmest kunne aflæse fejlkoden.

På baggrund af dette kan der opstilles et par kerneproblemer, der skal løses.

1. Hvordan fungerer det hexadecimale talsystem?
2. Hvordan skal brugeren opnå forståelse for det hexadecimale system?
3. Hvordan skal brugeren have mulighed for at foretage en konvertering?
4. Hvordan skal konverteringen af tal og bogstaver ske?

På baggrund af ovenstående analyse og kerneproblemer har vi fremstillet en problemformulering, som ses i næste kapitel.

## 1.4. Problemformulering

I projektet skal der opstilles en problemformulering, der behandler det overordnede emne og involverer kerne problemerne.

Her ses vores problemformulering:

I mange digitale systemer angives en fejlkode med et hexadecimalt tal, disse kan være uoverskuelige både at forstå og regne på. Kan gruppen fremstille et læringsprodukt, der kan konvertere disse tal til decimaltal fra 10-talssystemet?

Denne problemformulering skal undersøges nærmere, hvorefter der skal opstilles et løsningsforslag til problemet, som der senere hen i rapporten vil kunne ses.

Vi vil udvikle en prototype, hvor brugeren kan få en forklaring på, hvad det hexadecimale system, hvad vi forstår ved et talsystem, og hvordan man konverterer fra hexadecimal til decimal (10-talssystemet)

## 2. Teori

### 2.1. Indledende afsnit

I dette kapitel vil indholdet fokusere på teori omkring diverse emner fra gruppens valg af målgruppe, værktøjer: programmeringssprog og udviklingsmiljøer. De tilhørende licenser findes i bilag.

### 2.2. Specifikation af målgruppe

I projektet vil gruppen primært fokusere på andre IT-interesserede unge, men udelukker ikke modtagere af højere alder. Af den grund bliver der i de indledende iterationer af produktet lagt særlig fokus på forklaring og anvendelse af produktet. Da vi har med talsystemer at gøre i projektet stilles der et specifikt krav til målgruppen, der muligvis indsnævrer målgruppen; de skal have matematik sans. Med dette menes en evne til at opfatte matematik i et eksempel og benytte den i en praktisk sammenhæng.

Når der derfor nævnes at målgruppen er unge menes der altså ikke unger nede i 3. klasse, men som minimum går i 1. G eller tilsvarende. Med andre ord er målgruppen altså gymnasieelever og derefter.

### 2.3. Konvertering

\* Delvis af dette afsnit er skrevet i Programmering C

I dette afsnit vil der blive lagt særlig fokus på, hvad et talsystem er og i hvilken sammenhæng de egentlig benyttes.

I hverdagen benytter de fleste af os kun 10-talssystemet. De fleste af os tænker ikke yderligere over hvordan et tal skal opfattes fordi "det er jo bare sådan det er". Og dette argument bærer også en vis sandhed. Men det er ikke indtil man bevæger sig længere ud i det matematiske univers og støder på andre talsystemer.

Et talsystem eller et notationssystem er en måde at repræsentere matematiske tal med. Et ciffer er et taltegn, hvor tallet '42' altså består af de to cifre "4" og "2".

Da det tal er skrevet i decimalsystemet, står der egentlig følgende  $4 \cdot 10^1 + 2 \cdot 10^0$  - dette er så lig 42 og repræsenteres blot sådan. Derved kan vi altså opskrive følgende for talsystemet med basen 10:  $TALLET\ HER = x \cdot 10^n + y \cdot 10^{n-1} + z \cdot 10^0$

Tallet der står længst til højre har altså eksponenten 0, 0. position, og derved giver  $4 \cdot 10^0$  altså blot 4.

Man kan også udtrykke  $42_{10}$  i andre talsystemer, eksempelvis med romertal.

Der er 42 "XXXXII".

Der er også et system som kun består af to værdier: det binære talsystem, også kaldt totalssystemet som kun består af 1 og 0. Det binære talsystem bruges blandt andet ved lagring af data på medier som hulkort/-strimmel, magnetbånd, eksempelvis DAT, magnetstribe på ID-kort, DVD og harddisk. Princippet opbygning er ens med titalssystemet, idet der kan være ét ciffer fra 0-9 på hver plads, men i det binære talsystem kan der kun og netop kun være ét af to ciffer, herunder 0 eller 1. I det binære talsystem læses der fra højre mod venstre, hvilket vil sige hvis der skal skrives 10, så er det binære tal lig med 1010. Binære talsystem har altså også ligeledes formelen  $x \cdot 2^n + y \cdot 2^{n-1} + \dots z \cdot 2^0 = TAL$

Da der kun kan antages to værdier, er det ret simpelt at skrive tal såsom  $(2)_{10} : (10)_2$  eller 14, så er det 1110, 15 er 1111 osv. Denne rækkefølge kan ses nedenstående:

1024	512	256	128	64	32	16	8	4	2	1
1	1	1	1	1	0	0	1	0	1	0

Det hexadecimale talsystem er baseret på basen 16. De ekstra cifre udover (0-9) udgøres af bogstaverne A til F. Dette kan ses nedenstående:



$0_{\text{hex}} = 0_{\text{dec}} = 0_{\text{oct}}$	0	0	0	0
$1_{\text{hex}} = 1_{\text{dec}} = 1_{\text{oct}}$	0	0	0	1
$2_{\text{hex}} = 2_{\text{dec}} = 2_{\text{oct}}$	0	0	1	0
$3_{\text{hex}} = 3_{\text{dec}} = 3_{\text{oct}}$	0	0	1	1
$4_{\text{hex}} = 4_{\text{dec}} = 4_{\text{oct}}$	0	1	0	0
$5_{\text{hex}} = 5_{\text{dec}} = 5_{\text{oct}}$	0	1	0	1
$6_{\text{hex}} = 6_{\text{dec}} = 6_{\text{oct}}$	0	1	1	0
$7_{\text{hex}} = 7_{\text{dec}} = 7_{\text{oct}}$	0	1	1	1
$8_{\text{hex}} = 8_{\text{dec}} = 10_{\text{oct}}$	1	0	0	0
$9_{\text{hex}} = 9_{\text{dec}} = 11_{\text{oct}}$	1	0	0	1
$A_{\text{hex}} = 10_{\text{dec}} = 12_{\text{oct}}$	1	0	1	0
$B_{\text{hex}} = 11_{\text{dec}} = 13_{\text{oct}}$	1	0	1	1
$C_{\text{hex}} = 12_{\text{dec}} = 14_{\text{oct}}$	1	1	0	0
$D_{\text{hex}} = 13_{\text{dec}} = 15_{\text{oct}}$	1	1	0	1
$E_{\text{hex}} = 14_{\text{dec}} = 16_{\text{oct}}$	1	1	1	0
$F_{\text{hex}} = 15_{\text{dec}} = 17_{\text{oct}}$	1	1	1	1

Hvis der eksempelvis vil regnes på talværdien af det hexadecimal AA1E gøres følgende:

$$10 \cdot 16^3 + 10 \cdot 16^2 + 1 \cdot 16 + 14 \cdot 1 = 40960 + 2560 + 16 + 14 = 43550$$

Mere præcist fås følgende:

$$AA1E_{16} = 43550_{10}$$

## 2.4. Valg af værktøjer

### 2.4.1. Programmeringssprog

Gruppen står til at vælge imellem to muligheder:

1. HTML med CSS og JavaScript
2. Python

HTML med CSS og JavaScript giver mulighed for at køre produktet i en webbrowser uden at foretage nogen reel form for konvertering af koden.

Med Python er det lidt en anden historie, med mindre man benytter sig af programmer såsom Glowsript.

Gruppen vælger HTML, CSS og JavaScript, da den har mest erfaring med dette, og produktet bliver derfor let tilgængeligt for målgruppen.

### 2.4.1.1. HTML

HTML står for Hyper Text Markup Language og er et meget omfattende markup sprog, der er standardiseret til at fremstille websider- og applikationer. Sammen med CSS (Cascade Styling Sheets) og JavaScript udgør HTML en vigtig hjørnesteen af kernefunktionaliteten af alt webtrafik. HTML er et front-end udviklingssprog idet alle tekster og medier som brugeren ender med at skulle tilgå enten er indeholdt eller refereret.

I HTML opbygges syntaksen af såkaldte 'tags' der skal åbnes og lukkes. Imellem disse tags kan der være tekst, links til billeder eller endda flere under-tags. I selve tagsne kan der indgå attributter, der eksempelvis fortæller om hvilken type det indledende 'tag' har.

Internetbrowseren modtager fra en webserver HTML-dokumentet som bliver fortolket og indholdet vises endeligt til brugeren. Herunder er vedhæftet den obligatoriske 'Hello World!' for HTML hvor den syntaks er vist:

```
<!DOCTYPE html>
<html>
  <head>
    <title>This is a title</title>
  </head>
  <body>
    <p>Hello world!</p>
  </body>
</html>2
```

### 2.4.1.2. CSS

CSS, Cascading Styling Sheets, er et sprog der benyttes til at beskrive det ønskede design af eksempelvis HTML/XHTML/XML dokumenter. Man kan indlejre style-kode i et HTML-tag hvis man blot skal give noget til et enkelt tag, som ikke skal gentages. Hvis en definition dog skal gælde for mange forskellige elementer i HTML-koden laver man en separat .css-fil og linker til denne i HTML-en med <link rel="stylesheet" href="FILE\_NAME.css". Alle definitioner der indgår i styling-filen vil derefter blive påført på elementer som enten matcher navn eller såkaldt 'class', som også er en attribut der kan tildeles et HTML-tag.

Vil man skrive noget style for eksempelvis et paragraf-tag kan det se således ud i styling-filen.

```
p {
  color: sienna;
  margin-left: 20px;
}
```

---

<sup>2</sup> Wikipedia - <https://en.wikipedia.org/wiki/HTML>

Vi ser altså, at de argumenter der skal gælde for elementet `p` skal være omringet af krøllede parenteser. Syntaksen herover viser også, at man benytter kolon, når man har skrevet f.eks. `font-family`, som refererer til hvilken font der skal benyttes. Når en linje afsluttes skal der sættes et semikolon så kompilatoren ved, at den skal gå videre til næste linje.

### 2.4.1.3. Javascript

JavaScript følger faktisk syntaksen fra CSS mere eller mindre identisk, så det behøver vi ikke fokusere på.

JavaScript er den del af en hjemmeside der tilføjer interaktivitet og selvom der begynder at være lidt interaktivitet indbygget i CSS og HTML kan man altså dynamisk indlæse en værdi fra et andet hjemmeside på kommando uden en eller anden form for JavaScript. I JavaScript er der, ligesom i eksempelvis Python og Java, mulighed for at importere et hav af biblioteker der udvider anvendeligheden af ens kode med en nærmest uendelig stor faktisk som blot bliver større jo mere kode der udvikles. Vi kommer ikke i dette projekt til at importere nogle libraries da der simpelthen ikke er brug for det.

## 2.5. Udviklingsmiljø

### 2.5.1. Indledende afsnit

I dette projekt er der gjort brug af forskellige værktøjer, der har hjulpet med fremstilling af koden og produktet, for hvilke der bliver gjort rede i dette afsnit.

### 2.5.2. GitHub

Et de overvejende essentielle værktøjer vi har gjort brug af i projektet er GitHub. Det gøres af den grund, at vi er to personer, der udvikler dette projekt. GitHub er en web-baseret hosting tjeneste for version control med Git, og benyttes primært til at opdatere, tracke og udvikle kode.



Hvis ikke gruppen brugte GitHub, skulle vi konstant sende hinanden besked med de nyeste ændringer i koden, den skal så manuelt flettes og opdateres. Dette problem elimineres fuldstændig og komplet af GitHub og deres Version Control. Version Control er en proces i deres system, der tjekker hver fil for de nyeste ændringer der har været foretaget og uploader disse til projektets “master-branch”, som altså er tilsvarende den gældende og nyeste version i et projekt. Skal man pludselig teste noget eksperimentel kode eller lave en meget udstrittende ændring i projektets kode er der mulighed for at lave en ny “branch” og synkronisere de nyeste ændringer til denne. Hvis der pludselig bliver enighed om at flette de to branches er der mulighed for dette 100%

automatisk, uden noget kode bliver slettet. Nogle funktioner bliver muligvis ødelagte, men dette er resultatet af en sådan sammenfletning.

Det er også muligt at tilgå en gammel version af en hvilken som helst branch der har været på et hvilket som helst tidspunkt i projektet, hvilket altså medfører at GitHub fungerer som logbog over projektet. Har man eksempelvis brug for at finde noget gammel kode frem er der ingen nødvendighed for at lave 25 filer for hver ændring der har været foretaget. Vores projekt kan findes på Github her<sup>3</sup>.

### 2.5.3. Brackets

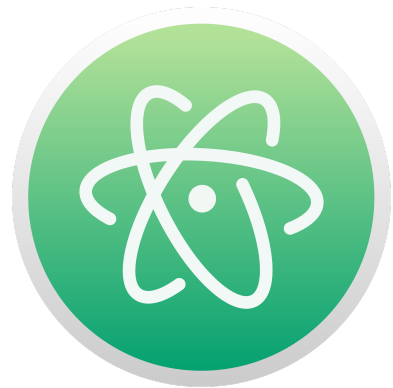
Brackets er en open-source samt cross-platform text-editor. Brackets er et udviklingsmiljø, der har fokus på de visuelle værktøjer og har mulighed for præprocessering. Brackets er et moderne tekstbehandlingsprogram, der gør det nemt at designe i browseren. Det er egnet for webdesignere og front-end udviklere. Brackets' hjemmeside kan findes her<sup>4</sup>.



Licensen til Bracket kan findes i bilag<sup>5</sup>.

### 2.5.4. Atom

Atom er også et open-source cross platform text editor, som i sin egentlighed blot består af HTML, JavaScript, CSS og Node.js. Det framework som Atom kører på er Electron, et lightweight framework der kan bruges til at lave applikationer på tværs af platforme ved brug af web-teknologier. Med Atom følger også naturlig integration af GitHub, så man hurtigt kan push/pull og sikre via version control at ens version er den nyeste. Der er også en indbygget package-manager således at man ligesom i Linux nemt kan tilføje, modificere og fjerne moduler og altså selve oplevelsen af Atom, hvilket sådan set giver næsten den samme frihed som i Linux.



Atom har den fordel, at brugeren hjælpes med automatisk fuldførelse og forslag til den kode man er i gang med at skrive (hvilket egentlig er en standardfunktion i mange populære text editors). Grænsefladen kan opdeles i flere ruder for at sammenligne og redigere kode på tværs af filer. Atoms hjemmeside findes her<sup>6</sup>.

---

<sup>3</sup> GitHub - [https://github.com/NathiNugget/Eksamensprojekt\\_konverteringer](https://github.com/NathiNugget/Eksamensprojekt_konverteringer)

<sup>4</sup> Brackets - <http://brackets.io/>

<sup>5</sup> Licenser - Punkt 7.3.1

<sup>6</sup> Atom - <https://atom.io/>

## 2.6. Laswells Kommunikationsmodel

Nedenstående ses Laswells kommunikationsmodel, som er en teoretisk model over hvordan en kommunikation mellem afsender og modtager kan forløbe. Der findes en række forskellige modeller, men dog er denne model meget god, idet modellen indkredser relevante punkter. Herunder er der tale om følgende punkter:

**Hvem:**

- Afsenderen i projektet er Ferhat og Nathaniel, som er forfatterne af rapporten.

**Hvad:**

- Konvertering mellem talsystemer er ikke særlig svært, når først man har styr på de systemer, konverteringen skal foregå imellem. Det eneste der skal til er kun og netop kun "Læren om at lære".

**Gennem hvilken kanal:**

- Formidlingen sker gennem vores rapport, hvor man kan lære forståelsen bag tal konvertering samt produktet - websiden.

**Til hvem:**

- Som sagt er vores målgruppe måske lidt bred, da vi primært fokuserer på unge med interesse for at rette fejl i digitale systemer - heraf interessen for tal konvertering. Derfor er vores målgruppe meget bred, og derfor er den egentlig kun afhængig af lysten til at lære om talsystemer og skelen imellem dem, læren om talsystemer og interessen i at rette fejl er begrænset til en specifik målgruppe.

**Med hvilken effekt:**

- Der er tale om en masse effekter i forhold til vores budskab. Blandt andet så tager nogle det som en form for udfordring, hvor man rent faktisk har fået lært noget i forhold til det faglige indhold. Ligeledes kan nogen mene, at det er spild af tid.

## 2.7. User Story

En User Story er en kort og simpel beskrivelse af opnåelse af et problem. User Stories skrives som regel ud fra en brugers synspunkt, som mere eller mindre ønsker en eller anden ny funktionalitet fra et givent IT-system.

---

<sup>7</sup> Licenser - Punkt 7.3.2

Herunder taler man om følgende 3 områder:

1. Hvem - som "rolle"
2. Hvad - ønsker jeg at "ønske"
3. Hvorfor/fordi - med det formål at "formål"

### 2.7.1. User story 1

Nedenstående ses et user story:

1. Hvem - Jeg er en elev på HTX-Roskilde.
2. Hvad - Som gerne vil lære, om hvordan systemudvikling fungerer på en simpel men informativ måde.
3. Hvorfor/fordi - Fordi jeg gerne vil bruge det i min nærmere fremtid.

### 2.7.2. User story 2

Nedenstående ses et user story:

1. Hvem - Jeg er en lærer på HTX-Roskilde.
2. Hvad - Som gerne vil lære mine elever om udvikling af et bruger interaktivt system i Javascript.
3. Hvorfor/fordi - Fordi jeg gerne vil bruge det i min nærmere fremtid.

### 2.7.3. User story 3

Nedenstående ses et user story:

1. Hvem - Jeg er elev på Roskilde Gymnasium.
2. Hvad - Som gerne vil lære om kode udviklingen af hjemmesider.
3. Hvorfor/fordi - Fordi jeg gerne vil programmere i HTML, CSS, JavaScript.

## 2.8. Usability

Jakob Nielsen definerer begrebet usability gennem 5 komponents kvaliteter, som kan ses nedenstående:

- **Learnability:** How easy is it for users to accomplish basic tasks the first time they encounter the design?

- **Efficiency:** Once users have learned the design, how quickly can they perform tasks?
- **Memorability:** When users return to the design after a period of not using it, how easily can they reestablish proficiency?
- **Errors:** How many errors do users make, how severe are these errors, and how easily can they recover from the errors?
- **Satisfaction:** How pleasant is it to use the design?

Jakob Nielsens 25 spørgsmål er meget gode, idet de giver et godt overblik over kommunikationen mellem modtager og afsender. En række af dem kan ses nedenstående:

1. Hvem er målgruppen?
2. Hvad er budskabet?
3. Hvad er mediet?
4. Hvilken effekt skal produktet have hos målgruppen?
5. Hvad er formålet med effekten hos målgruppe?
6. Hvem er afsenderen?
7. Hvilken effekt skal produktet have hos afsenderen?
8. Hvad er formålet med effekten hos målgruppen?
9. Hvordan påvirkes målgruppen ellers lignende budskaber?
10. Er produktet lavet før?
11. Hvor, hvornår, hvordan skal målgruppen opleve produktet?
12. Hvordan skal produktet distribueres?
13. Hvilken genre skal bruges?
14. Hvilken fortællemåde skal bruges?

Et par af disse spørgsmål er besvaret gennem Laswells kommunikationsmodel, som vi tidligere har været igennem.

## 2.9. Extreme programming

Extreme Programming projektet blev startet i 1996 den 6. marts. Siden da har metoden vist sig at være meget succesfuld blandt både mindre og større virksomheder. Extreme Programming metoden er god at benytte, da den er fikseret på modtagerens tilfredshed. I stedet for at levere så meget kode som muligt en eller anden dato langt ude i fremtiden leverer den bider når der er brug for det. Det leder til at man kan få feedback og kan medtage denne feedback i planlægningen af næste iteration/videreudvikling af koden. Metoden lægger fokus på teamwork, således bliver alle medarbejdere ligestillet uanset hvilken sektor eller fokuspunkt de lige præcis arbejder på. Metoden er brugt til at udvikle systemudviklingsmetoden på HTX → Roskilde, og vi benytter derfor delvist og indirekte denne metode i vores udvikling af produktet. Extreme Programming kan findes her<sup>8</sup>.



---

<sup>8</sup> Extreme Programming - <http://www.extremeprogramming.org/>

## 2.10. OSI-modellen

OSI-modellen er en model, der benyttes til at beskrive netværskommunikation, hvilket der egentlig er en smule af i vores projekt.

Lag	Elementer
7 - Applikationslaget Danner grundlag for at brugeren har adgang til information på nettet via. programmer. Dette er brugergrænsefladen	HTML-side, CSS og JavaScript.
6 - Præsentationslaget Omdanner data til en (for et program) kendt standardgrænseflade, og/eller andre datastrukturer	HTTP
5 - Sessionslaget En del af TCP	Webserver
4 - Transportlaget Tillader transport af pakker imellem modtager og afsender. Disse kan være "state-" og "connection-" orienterede og sikrer derved afsending af pakker der aldrig kom frem til modtageren.	TCP
3 - Netværkslaget Udfører routing-funktioner, sørger for at sende pakker til rette modtager, og kan udføre ind- og udpakning om rapportere om leveringsfejl	Routing table.
2 - Data Link-laget Her gives mulighed for at overføre data mellem netværks-moduler og finde, muligvis rette, fejl der kan optræde i det fysiske lag.	MAC-adressen, 2.4GHz WiFi.
1 - Fysisk lag På dette lag tages der hensyn til alle fysiske og elektriske rammer hvorved kommunikation foregår	Oprettelse og afslutning af elektronisk forbindelse til overførsel modul. Dataflow i mellem kabler og repræsentationsmedie



### 3. Iterationer af system udviklingsaktiviteter

#### 3.1. Indledende afsnit

Produktudviklingen i dette projekt dannes af iterativ systemudvikling som den fremgår af systemudviklingsmetoden fra HTX → Roskilde. I projektet udarbejdes tre iterationer af produktet således at kernefunktionalitet kan verificeres som funktionel.

Link til uddybende forklaring af systemudviklingsmetoden kan findes her<sup>9</sup>.

<https://www.lectio.dk/lectio/523/ExerciseFileGet.aspx?type=opgavedef&exercisefileid=33362142248>

#### 3.2. Første iteration af systemudviklings-aktiviteterne

##### 3.2.1. Planlægning

For denne iteration skal vi blot opstille en spike-solution til hvordan det endelige produkt kommer til at fungere. Det vil sige vi skal opstille en HTML-side, som indeholder et input-felt og en knap der skal eksekvere en funktion. For denne funktion skal der naturligvis også indgå en fil med JavaScript.

##### 3.2.2. Design

Når vi kigger lidt på hvad der kommer til at skulle fremstilles er det egentlig blot et standard HTML-dokument med et input-felt og en knap, der starter en funktion. Denne funktion skal så findes i en JavaScript-fil.

Da disse opgaver ikke er omfattende, behøver vi ikke filosofere meget over, hvordan vi fremstiller.

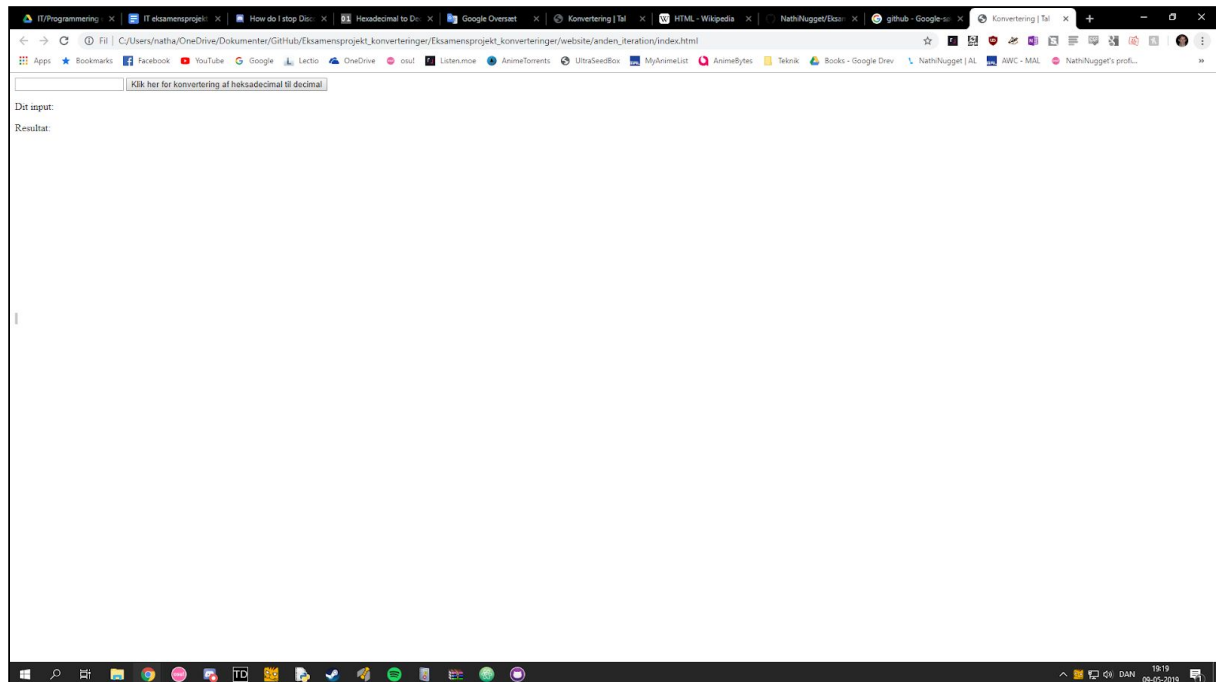
##### 3.2.3. Brugergrænseflade

Brugerfladen i denne iteration er så simpel som den overhovedet bliver, der kommer til at være et inputfelt og en knap. Et billede af hvordan det kommer til at se ud

---

<sup>9</sup> Systemudviklingsmetoden -

<https://www.lectio.dk/lectio/523/ExerciseFileGet.aspx?type=opgavedef&exercisefileid=33362142248>



Hvis det ikke er tydeligt på billedet er der oppe i venstre hjørne 1 inputfelt og en knap. Nedenunder disse er der to paragraffer der ikke bruges til noget i denne iteration.

### 3.2.4. Programmets virkemåde

Programmet opbygges af en HTML-side der indeholder et input-felt med typen text og en knap til højre der eksekverer funktionen yeet().

Vi har nedenstående lavet et skema for de tags, udtryk og funktioner der indgår i henholdsvis HTML og JavaScript.

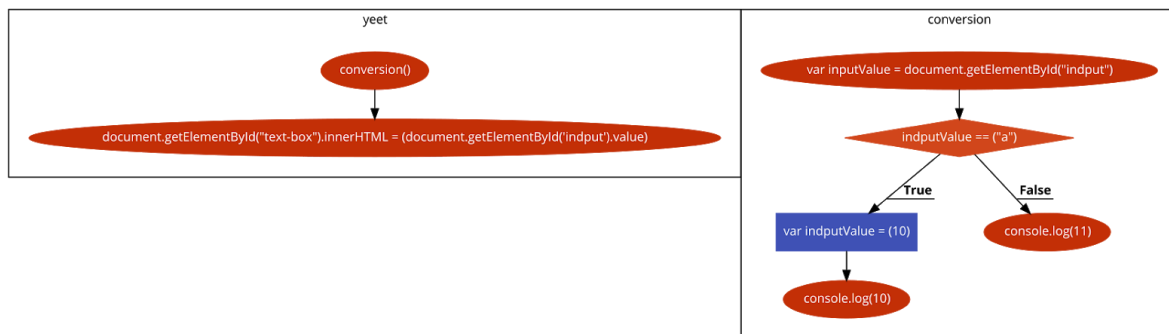
Tag	Forklaring
<!DOCTYPE HTML>	Fortæller browseren, at dette browseren at dette er et HTML-dokument. Da ingen parametre er vedhæftet benyttes den nyeste version (HTML5)
<html>	Her starter selve HTML'en.
<head>	Her defineres hvilket sprog indholdet er, hvilket tegnsæt der skal understøttes på siden mm. Dette er altså en slags opsætning.
<body>	Her skal alt indhold være.
<h1>	Benyttes til overskrifter, kan også skrives mindre som <h2>, <h3> osv.

<code>&lt;p&gt;</code>	Paragraf-tag til tekst.
<code>&lt;br&gt;</code>	'Break'-tag der bruges til linjeskift.
<code>&lt;input type="text" id="indput"&gt;</code>	Laver et inputfelt, markeret som tekst og givet id'et 'indput'.
<code>&lt;button onclick="yeet()"&gt;</code>	En knap, der afvikler funktionen 'yeet()', denne er defineret i 'website.js'.

Resultatet af alt dette fremgår af billedet i brugergrænseflade

Da der ikke som sådan er nogen funktionsloop kommer vi ikke til at lave et flowchart over HMTL-koden.

For lige at få en idé om hvordan vi skal organisere koden i JavaScript fremstiller vi lige et flowchart.



Ligeledes som for skemaet herover laver vi også lige et funktionsskema for vores JavaScript som i denne iteration blot er en hardcoded konvertering af et enkelt ciffer.

Funktioner & udtryk	Forklaring
<code>console.log("TEXT_HERE");</code>	Printer en værdi eller noget tekst til konsollen. I tilfældet til højre printes "TEXT_HERE", men kan erstattes af hvilken som helst string eller variabel.
<code>var convert = true;</code>	Definerer variabelen convert og sætter denne til true. Den benyttes muligvis til en senere iteration for at tjekke om en værdi er blevet konverteret.
<code>function yeet(){}</code>	Funktionen der køres når der klikkes på knappen. Imellem de krøllede parenteser indeholdes nedenstående funktioner.

<pre>var short = document.getElementById('result');</pre>	<p>Laver et kortere udtryk for at referere til den paragraf, der skal indeholde resultatet.</p>
<pre>if (convert == true){}</pre>	<p>Hvis dette er sandt, køres nedenstående kode.</p>
<pre>console.log("yeet");</pre>	<p>Skriver yeet til konsollen, blot som test for at boolean-udtrykket ovenfor er sandt.</p>
<pre>if (inputValue === "A");</pre>	<p>Hvis inputValue har samme type og værdi som bogstavet 'A', køres følgende kode.</p>

Herunder skrives koden som den står for denne iteration.

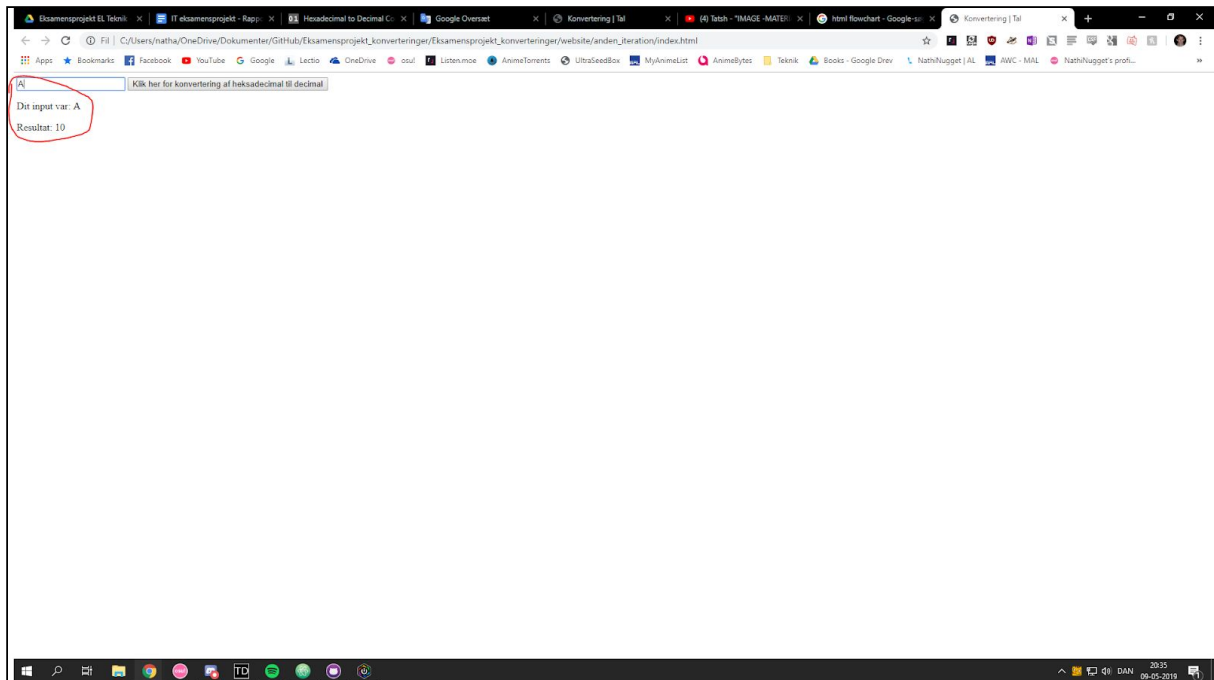
Den fulde kode for denne iteration kan findes i bilag

The screenshot shows the Atom code editor with two files open: index.html and index.js. The index.html file contains HTML code for a calculator interface, including a title, a display, and buttons for digits, operations, and conversion. The index.js file contains JavaScript code for the calculator's logic, including a function to convert a decimal number to a fraction and a function to convert a fraction to a decimal number. The code is written in a clean, readable style with comments in Danish.

Her ses hvordan der let arbejdes på tværs af filer, med HTML/CSS i Atom.

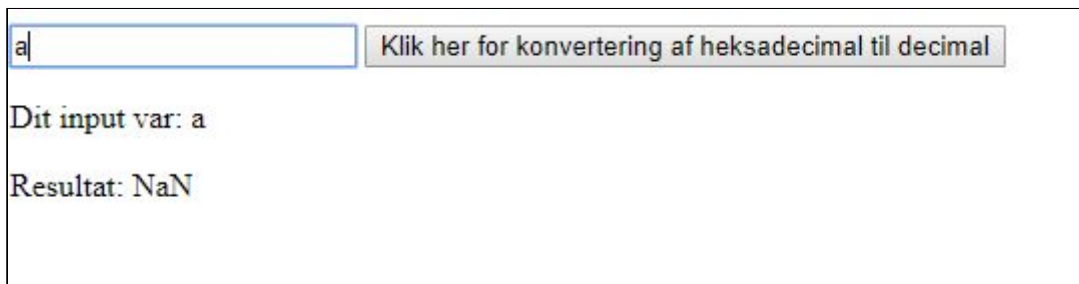
### 3.2.5. Test

Ved test af første iteration tester vi for den eneste værdi vi har hardcoded ind - "A"



Vi ser hvordan resultatet bliver 10 decimal.

Tester vi med andre værdier, så som lille a, ser vi dog følgende.



Resultatet er NaN.

### 3.2.6. Resultatopgørelse

Vi har oprettet udviklet produktet i en iteration, som stadig mangler kernefunktionalitet. Men vi har løst et kerneproblem: at lave et website med mulighed for brugerinput. Det betyder at der i næste iteration stadig skal implementeres den faktiske kernefunktionalitet.

## 3.3. 2. Iterations af systemudvikling

### 3.3.1. Planlægning

I denne iteration skal vi videreudvikle på første iteration. Der skal sikres en funktion, der rent faktisk kan konvertere et hexadecimalt tal til et decimaltal.

### 3.3.2. Kontrol af krav og testproducerer

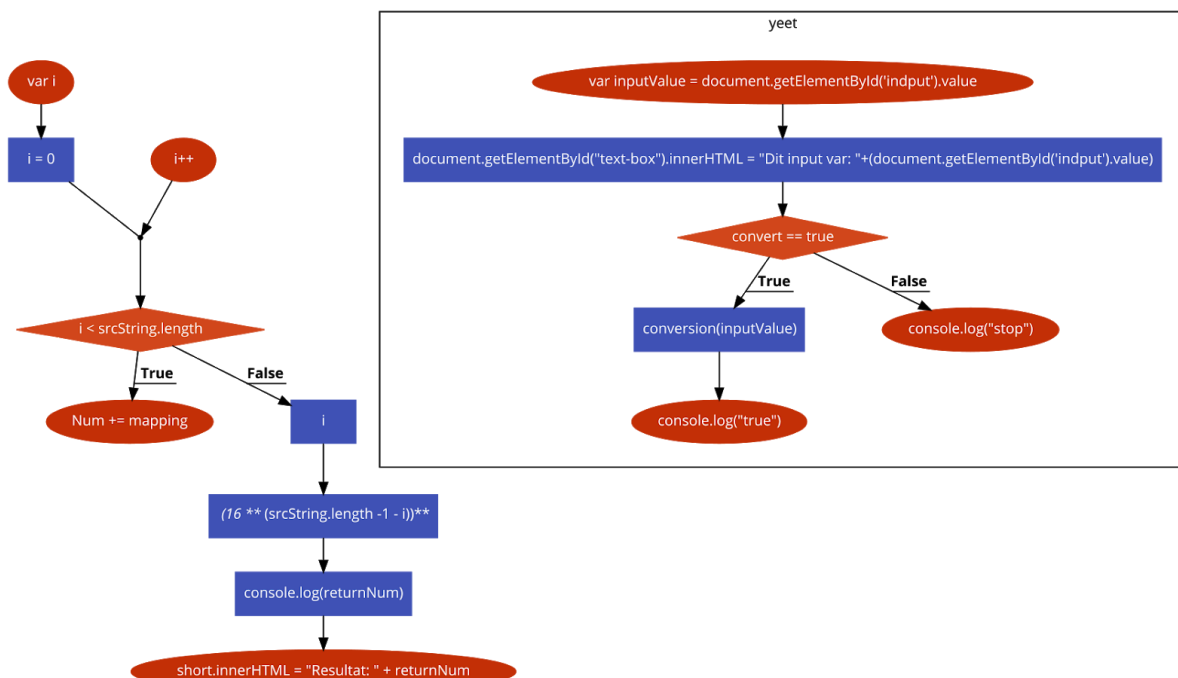
For denne iteration skal den faktiske konvertering ske, det betyder altså, at vi skal finde ud af hvordan vi kan tage et hexadecimalt input og konvertere det. I testen skal der testes for tegn der medhører de hexadecimale værdier, men også tegn som ikke indgår i hexadecimale tal, såsom lille a osv.

### 3.3.3. Design

For denne iteration ændrer HTML'en sig ikke, og vi medtager derfor ikke funktionsskemaet for denne iteration. Designet ændrer sig heller ikke det store, men vi kan lige lave et flowchart for hvordan konverteringen skal foregå.

1. Input skal gemmes som variabel
2. Der skal eksekveres en ny funktion der tager input som parameter
3. Denne nye funktion skal tvinge input til at gemmes som string så den kan indekseres.
4. Der skal skrives decimale værdier til enhver hexadecimal værdi
5. Der skal laves et for-loop i hele tiden bliver større.
6. Værdien af index[i] af input skal ganges med 16 opløftet i længden af input minus i minus 1. Grunden til at der også skal trækkes 1 fra er fordi indeksering starter ved 0.
7. Denne værdi gemmes og adderes til en variabel hver gang
8. Værdien skrives til tekstboksen på siden og printes i konsollen.

Så dannes flowchart over dette:



### 3.3.4. Implementering

Ved implementation Så opskriver vi lige funktionsskema over funktioner der kommer til at blive indhold i JavaScript-filen.

\* Dele af dette skema er udviklet i Programmering C

Funktioner	Forklaring
<pre>var mapping = {"A": "10", "B": "11", "C": "12"};</pre>	Liste over hvad et hexadecimalt tal er i decimal, hvor der i listen til venstre er tastet tre værdier og deres tilsvarende værdier fra listen.
<pre>var srcString = inputValue.toString();</pre>	Variablen srcString tillægges data fra inputValue og gemmes som string, altså tekst. Dette muliggør indeksering over denne, så vi kan kigge på hver enkelt
<pre>var returnNum = 0;</pre>	En variabel for summen af alle tal efter konvertering
<pre>var i;</pre>	En variabel der blot skal bruges i en for-løkke
<pre>for (i = 0; i &lt; srcString.length; i++){}</pre>	Løkken der starter konverteringen. Den tager variabelen i og sætter den lig 0 som parameter. Løkken eksekveres igen og igen så længe i er mindre end længden på srcString. Til sidst tillægges i 1.
<pre>returnNum += mapping[srcString[i]] · (16 · · (srcString.length - 1 - i));</pre>	Variablen returnNum tillægges hele tiden værdien af beregningen til højre. værdien af den decimale værdi
<pre>short.innerHTML = "Resultat: " + returnNum;</pre>	Skriver den konverterede værdi til returnNum

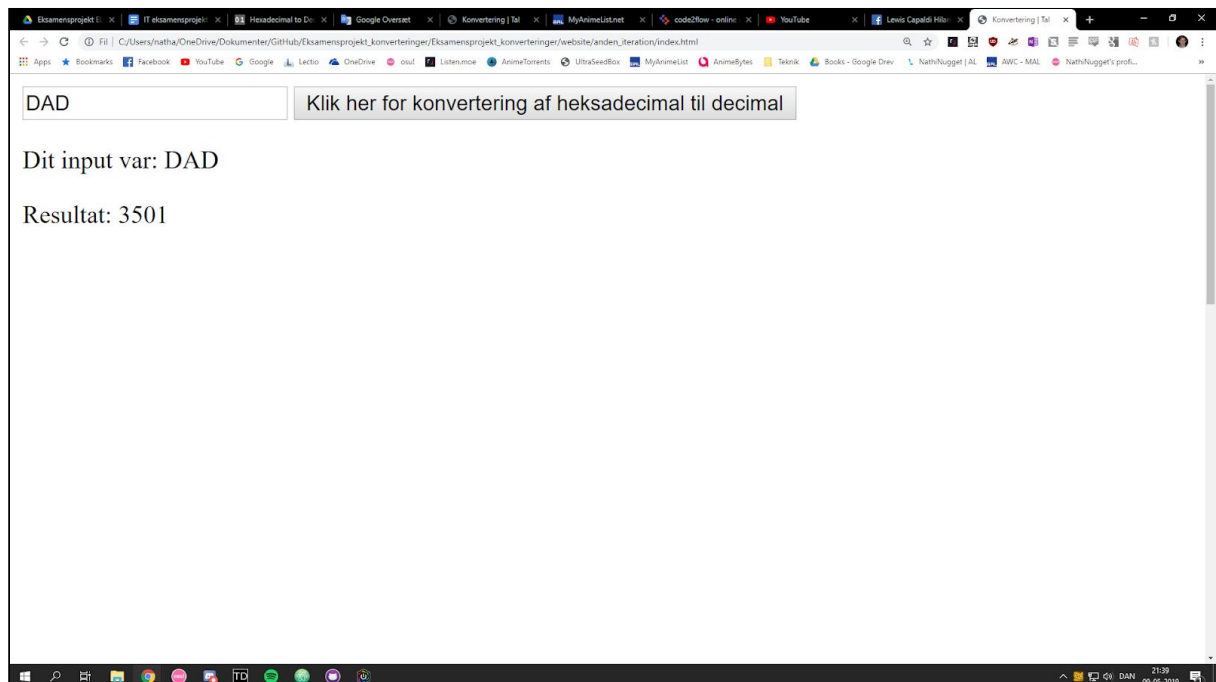
Screenshot over koden kan ses nedenstående:

The screenshot shows a code editor with two files: `index.html` and `website.js`. The `index.html` file contains a basic HTML structure with a `body` class and a `main` container. The `website.js` file contains a `conversion` function that takes an input value, converts it to a decimal value, and returns the result. The function uses a `for` loop to iterate over the input string and calculate the decimal value.

Nu skal vi egentlig blot til testen af koden.

### 3.3.5. Test

Ved test af koden ses følgende resultater.



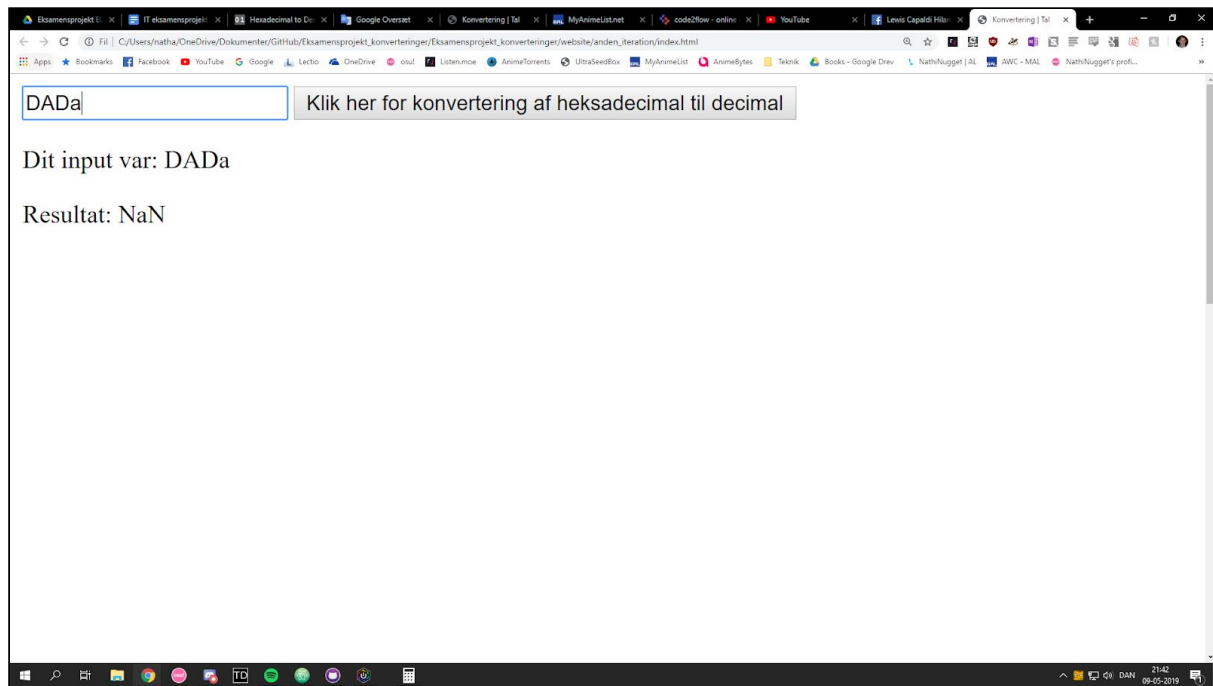
Det ses, at når input er DAD bliver resultatet 3501. Men passer det?

$$13 \cdot 16^2 + 10 \cdot 16^1 + 13 \cdot 16^0 = 3328 + 160 + 13 = 3501$$

Resultatet passer altså, og konverteringen fungerer altså.



Hvad med hvis vi indsætter lille 'a'?



Resultat bliver NaN, da a ikke er defineret i listen og blandt andet heller ikke har en talvædi, derfor gives fejl i vores for-loop, så der er altså fejl, hvis tegn uden for det hexadecimale talsystem indgår i input.

### 3.3.6. Resultatopgørelse

Vi har for denne iteration oprettet en liste over hexadecimale værdier og hvilke værdier der er de tilsvarende i decimalsystemet. Derved resulterer dette i, at vi kan bruge input som string og indeksere denne: vi kan benytte dette i en funktion og oversætte den matematiske funktion for et hexadecimalt tal. For denne iteration er et andet af vores kerneproblemer altså løst: at få foretaget en konvertering af et hexadecimalt tal til decimal. I næste og sidste iteration som vi også kalder prototypen kommer den faktiske grafiske brugerflade, forklaring af det hexadecimale system for brugeren og også konvertering af binær til decimal.

## 3.4. 3. Iteration af systemudviklingsaktiviteterne/prototype

### 3.4.1. Planlægning

I denne iteration videreudvikles anden iteration, hvor den egentlige konvertering af hexadecimale værdier til decimal blev introduceret, nu mangler blot forklaring af det hexadecimale system for brugeren. Ydermere introducerer vi også konvertering fra binær til decimal i prototypen. Der vil også blive udviklet noget grafisk design i denne model, og det er her CSS kommer ind i billedet.

### 3.4.2. Kontrol af krav og testproducerer

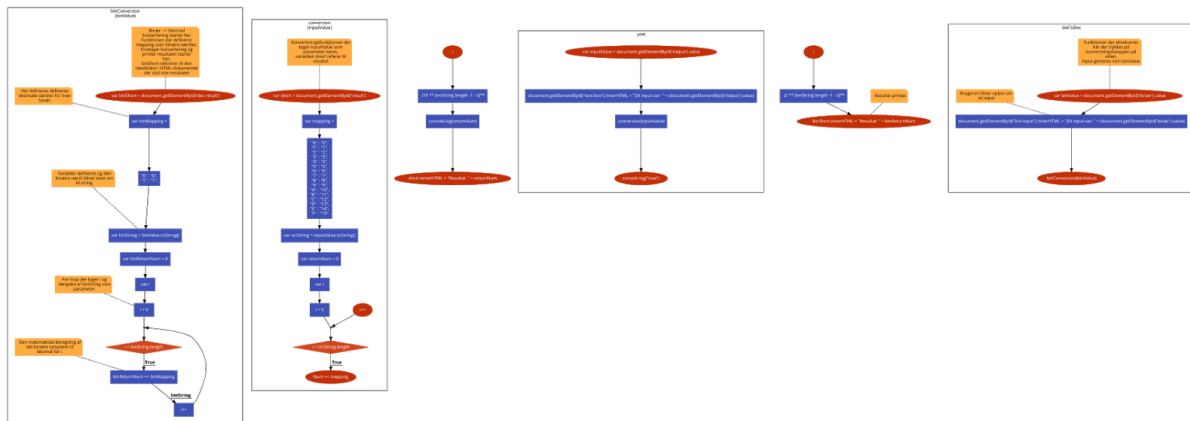
Nedenstående ses en række krav til tredje iteration af systemudviklingsaktiviteterne:

1. Binær til decimal konvertering skal implementeres.
2. Det hexadecimale talsystem skal forklares .
3. Brugergrænseflade skal designses.

### 3.4.3. Design

Nedenstående ses et flowchart over tredje iteration:

Denne iteration indebærer egentlig blot

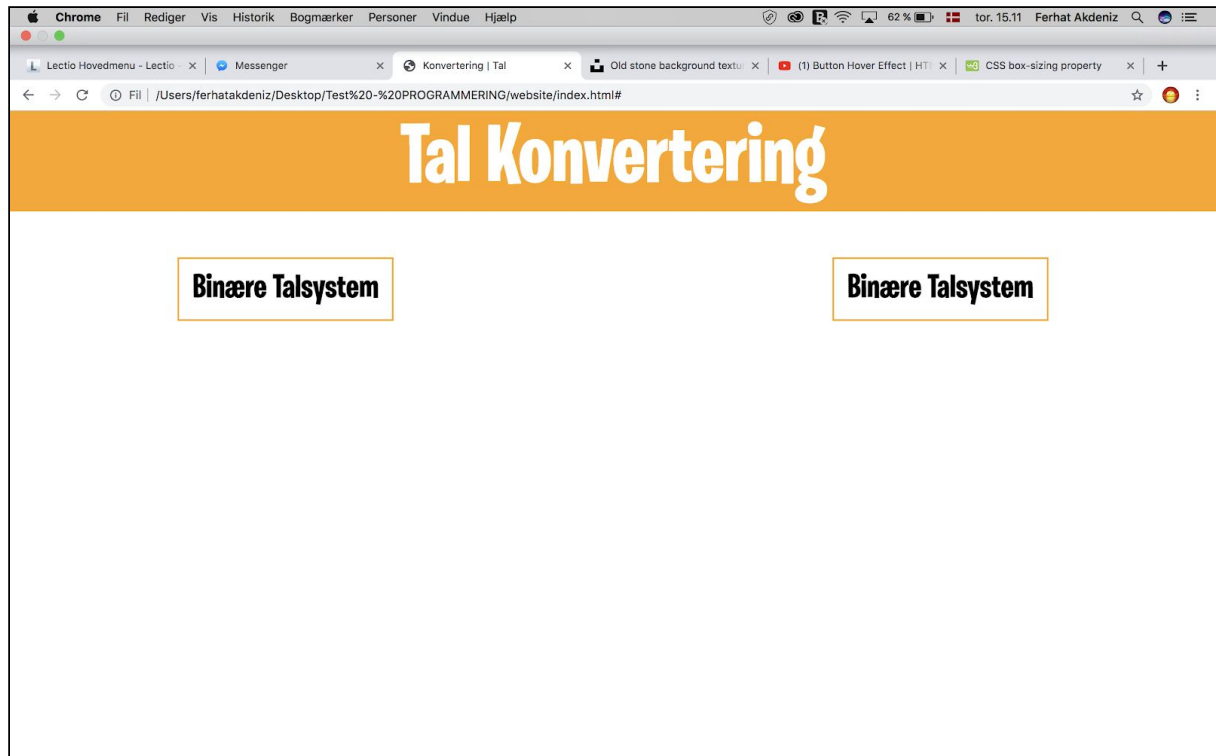


Ovenstående flowchart kan også findes i bilag med større opløsning<sup>10</sup>.

### 3.4.4. Implementering

I sidste iteration lægges der vægt på det sidste og selve designet på hjemmesiden.

Nedenstående ses et billede af opbygningen af hjemmesiden, som består af en topbar med tekst. Her benyttes både HTML og CSS, som kan ses nedenstående:



## HTML

1. `<div class="top_bar">`
2. `<h1 style="font-family:burbank big condensed">Tal Konvertering</h1>`
3. `</div>`

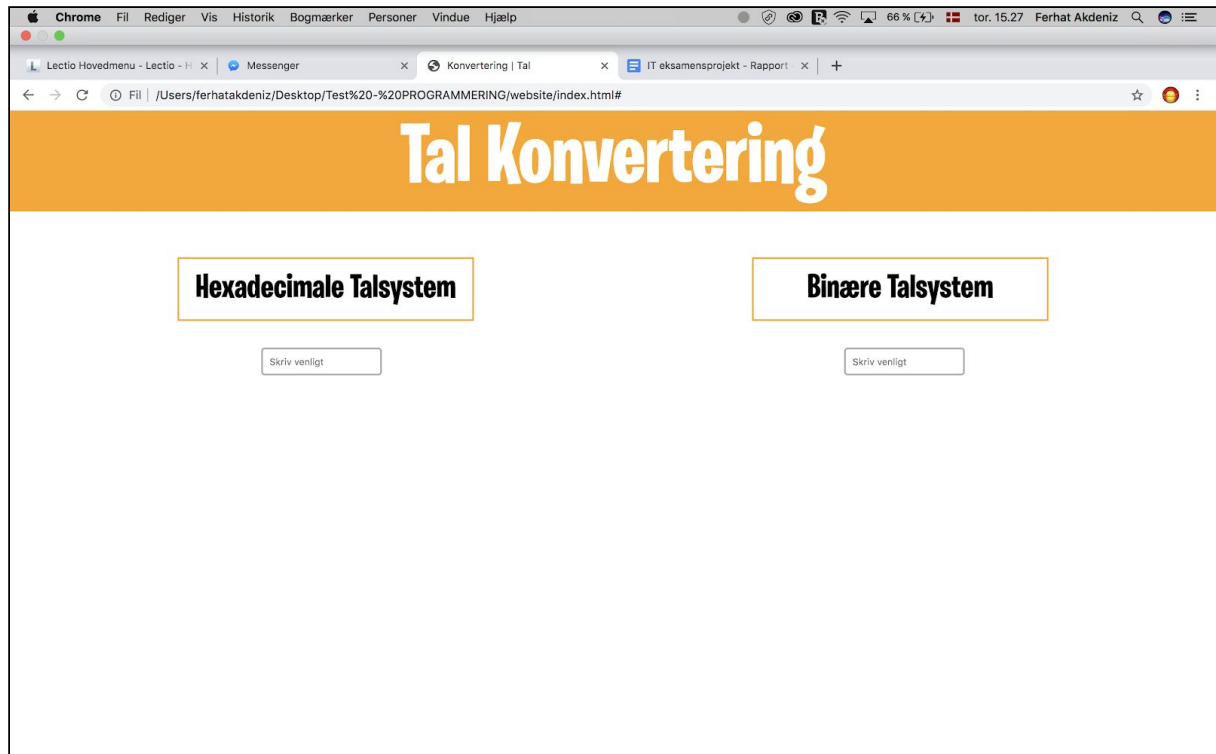
Ovenstående er teksten defineret med et div tag, og et h1 tag, som bruges til overskrifter.

## CSS

1. `.top_bar {`
2. `width: auto;`
3. `height: 120px;`
4. `background-color: orange;`
5. `color: black;`
6. `}`
- 7.

Ovenstående funktion gør, at top baren for en højde på 120 pixels med en orange baggrundsfarve. Width-taget angiver bredden på baren, som er givet auto i følgende tilfælde, så det fylder hele længden.

Næste trin er så, at få indsat 2 input felter, som giver brugeren lov til at indsætte den valgte talværdi for konverteringen.



## HTML

1. `<input type="text" id="input" onkeyup="yeet()" placeholder="Skriv venligst">`

Ovenstående tilføjer et input felt, hvor 'type' er den værdi der skal arbejdes med. Herunder er der valgt text, hvilket også kunne være numbers (tal), slider eller password bl.a.. ID'et angiver identifikation for et tag, således at man kan referere til det med f.eks. JavaScript og JQuery. `onkeyup="yeet()"` eksekverer en funktion så snart brugeren slipper tasten de trykkede på for at give input. Det vil sige, at hver gang man skriver et ciffer/tegn vil funktionen eksekveres. Til sidst er der `placeholder="Skriv venligst"` tagget, som indsætter teksten i input feltet. Denne tekst forsvinder så efter brugeren har klikket på input feltet.

## CSS

1. `input[type=text]{`
2. `width: 17%;`

Width funktionen angiver som sagt bredden for inputfeltet, som i dette tilfælde er sat til 17%. Ligeledes kunne det også skrives i pixels.

3. `border:2px solid #aaa;`

`border:2px solid #aaa;` giver brugeren mulighed for at angive stil, bredde og farve på et elements grænse (kanter). 2px er størrelsen på kanten, hvor solid betyder at linjen skal være solid, altså det skal eksempelvis ikke være prikkede linjer. #aaa angiver farven på kanten, som i dette tilfælde er grå.

4. `border-radius:4px;`

Border-radius:4px; angiver størrelsen på kantens radius, som i dette tilfælde er 4 pixels.

5. `margin:8px 0;`

margin:8px 0; bruges til at skabe plads omkring elementer uden for de definerede grænser.

6. `outline:none;`

outline:none; er en linje, der er trukket rundt om elementerne, uden for kanterne for at gøre elementet "stand out".

7. `padding:8px;`

padding:8px; bruges til at genere plads omkring et elements indhold, inden for alle definerede grænser (kanter).

8. `box-sizing:border-box;`

box-sizing:border-box; denne egenskab definerer hvordan bredden og højden af et element beregnes: skal de inkludere padding og borders, eller nej.

9. `transition:.3s;`

transistion:.3s; er overgangen, i følgende sammenhæng er den sat til 0,3 sekunder.

10. `position: absolute;`

position: absolute; specificerer typen af positions metoden for et element (static, relative, absolute, fixed eller sticky).

11. `top: 33%;`

12. `left: 13.9%;`

13. `}`

14.

top og left angiver hvor elementet skal placeres.

15. `input[type=text]:focus{`

16. `border-color:dodgerBlue;`

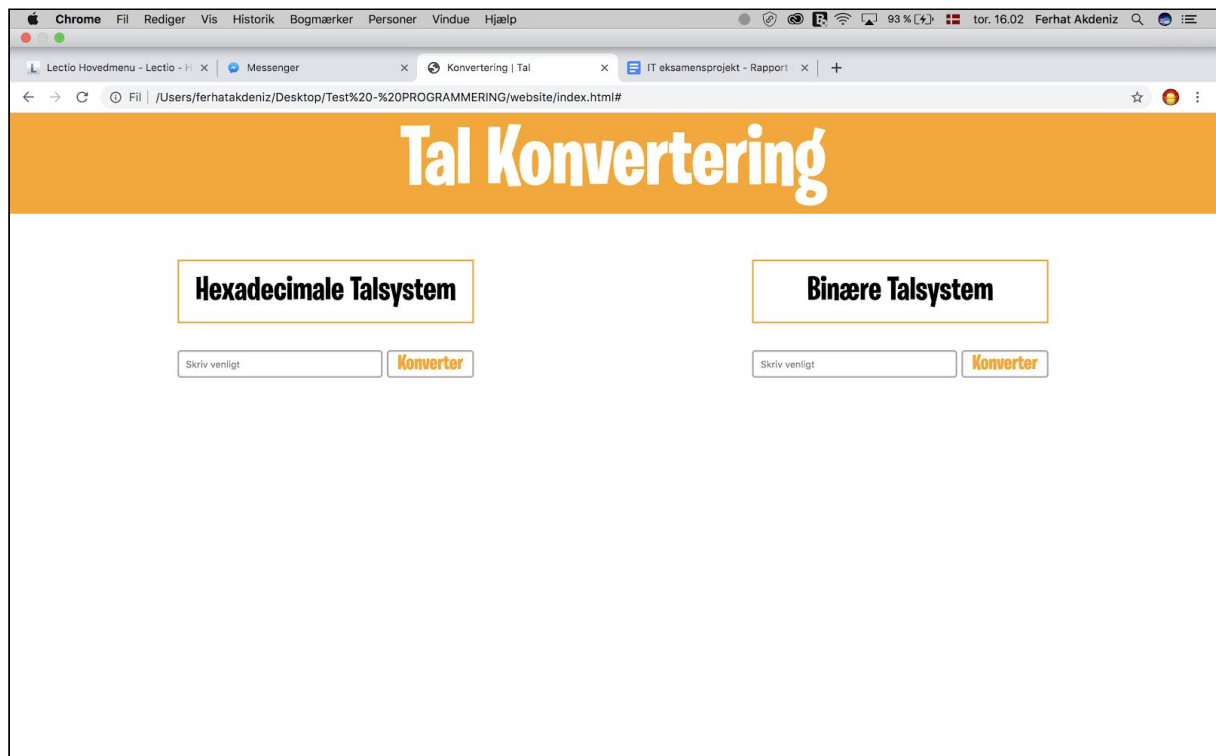
17. `box-shadow:0 0 8px 0 dodgerBlue;`

18.

Når brugeren klikker på input feltet vil der forekomme en blå effekt rundt omkring kanterne, hvilket gøres ved hjælp af ovenstående kode. dodgerBlue er farven, hvor 8px angiver størrelsen af denne effekt. Nedenstående ses det tydeligere:

Konverter

Næste trin er så at få integreret buttons, så handlingen i input feltet kan foretages.



## HTML

1. `<button onclick="yeet()" class="klik" style="font-family:burbank big condensed">Konverter</button>`

Ovenstående ses koden for knappen, hvor onclick="yeet()" opstår når brugeren klikker på knappen. Her bliver der refereret til yeet(). class="klik" angiver navnet for koden, som kan refereres til CSS, hvor koden kan ændres visuelt. style="font-family:burbank big condensed" er en skrifttype, som bruges i følgende sammenhæng.

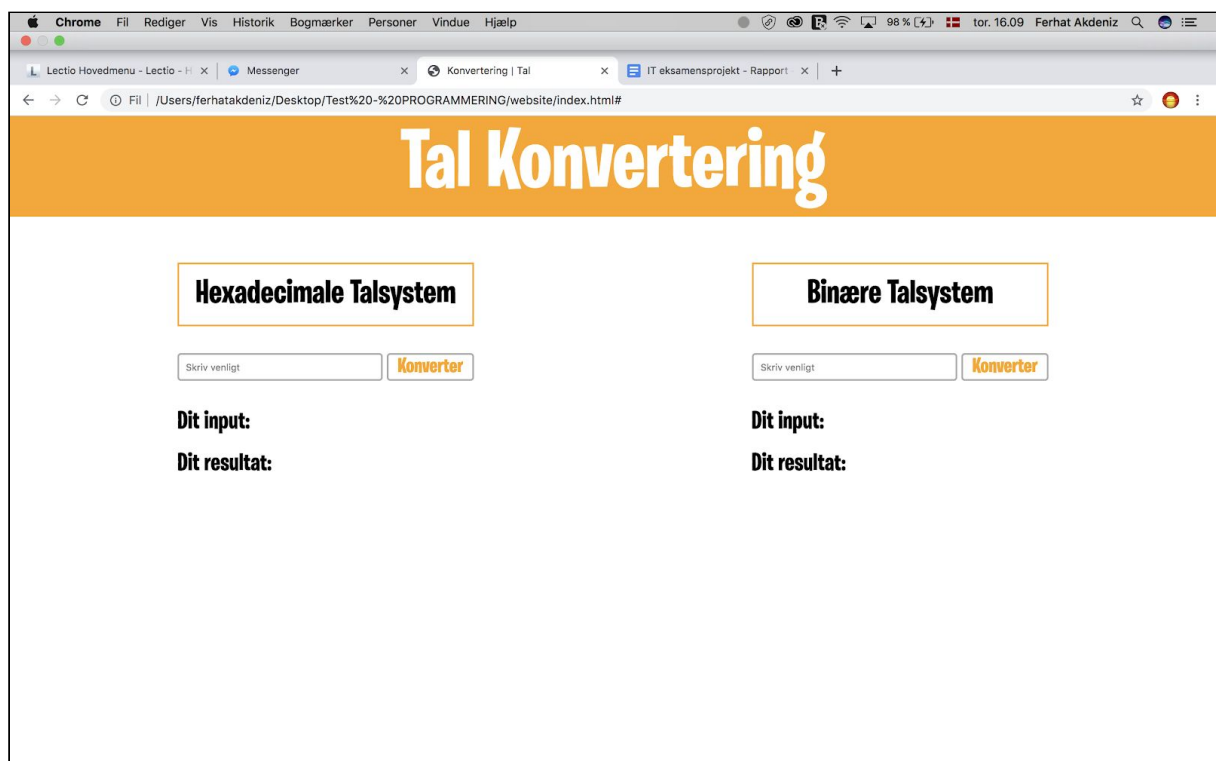
## CSS

1. `.klik {`
2. `position: absolute;`
3. `top: 34%;`
4. `left: 31.3%;`
5. `width: 7.3%;`
6. `height: 4%;`

7. `border: 2px solid #aaa;`
8. `border-radius: 4px;`
9. `color: orange;`
10. `font-size: 25px;`
- 11.

Ovenstående ses samme elementer som tidligere, hvor color og font size er nyt. color: orange; gør at teksten i knappen er orange, hvor font-size: 25px; angiver størrelsen på teksten, i følgende sammenhæng 25 pixels.

Næste trin er så, at vise konverteringen, som kan ses nedenstående. Både det indtastede og resultatet for konverteringen bliver vist her.



## HTML

1. `<p class="text-box" id="text-box" style="font-family:burbank big condensed"> Dit input: </p>`
- 2.
3. `<p class="resultat" id="result" style="font-family:burbank big condensed"> Dit resultat: </p>`
- 4.

Ovenstående ses koden, hvor elementerne er ens med forrige dokumenterede elementer.

## CSS

1. `.text-box {`

```
2.     position: absolute;
3.     top: 43%;
4.     left: 13.8%;
5.     font-size: 30px;
6. }
7.
8. .resultat {
9.     position: absolute;
10.    top: 50%;
11.    left: 13.8%;
12.    font-size: 30px;
13. }
14.
```

Her ses koden for CSS'en til teksten, hvor elementerne ligeledes er ens med forrige dokumenterede elementer.

Næste step er så, at give en forklaring på konverteringen. Herunder kort omkring talsystemet, og derefter en redegørelse for beregningen af konverteringen. Til sidst er en tabel opstillet, hvor sammenhængen mellem binære, decimale og hexadecimale kan ses. Da koden er lang sættes den ikke ind, men kan findes i bilag.

Binary Number	Decimal Number	Hex Number
0	0	0
1	1	1
10	2	2
11	3	3
100	4	4
101	5	5
110	6	6
111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F
10000	16	10
10001	17	11
10010	18	12
10011	19	13
10100	20	14
10101	21	15
10110	22	16
10111	23	17
11000	24	18
11001	25	19
11010	26	1A
11011	27	1B
11100	28	1C
11101	29	1D
11110	30	1E

**Hexadecimale Talsystem**

Skriv venligt

**Dit input:**

**Dit resultat:**

**Hvad er hexadecimal talsystem?**

I det talsystem vi til dagligdag benytter, har vi basen 10. Det betyder altså, at når man når over 9, begynder man forfra ved det første tal og sætter et 0 bag på. Så skriver vi altså 10. Det gør man ikke i det hexadecimal talsystem. I dette system har man i stedet basen, så man skriver tallene fra 0 og op til og med 15, eller det hedder i det hexadecimal talsystem "F". I dette system skriver man ikke 10, man det første bogstav i alfabetet og det næste indtil man når "F" - eller 15 i decimalsystemet/10-talsystemet. Herunder ses sammenligningen imellem de to, så man kan danne sig et lille overblik på hvordan de to systemer hænger sammen. For at omregne fra hexadecimal til decimal, læser man fra højre, ganger man symbolets værdi med 16 opløftet i symbolets position fra højre side. Det vil sige, står der f.eks. 1A hedder udregningen:  $1016 * 0 + 1616 * 1 = 26$ .

Et skema over denne konvertering kan ses på højre side.

**Binære Talsystem**

Skriv venligt

**Dit input:**

**Dit resultat:**

**Hvad er binær talsystem?**

Det binære talsystem, eller også kaldt totalssystemet som kun består af to cifre: 1 og 0. Det binære talsystem bruges blandt andet ved lagring af data på medier som hukort/-strimmel, magnetbånd, eksempelvis DAT, magnetstripe på ID-kort, DVD og harddisk. Princippet opbygning er ens med titalssystemet, idet der kan være et eller flere cifre på hver plads, men i det binære talsystem kan der kun og netop kun være ét af to cifre, herunder 0 eller 1. I det binære talsystem læses der fra højre mod venstre, hvilket vil sige hvis der skal skrives 10, så er det binære tal lig med 1010, eller 14, så er det 1110, 15 er 1111 osv.

For at opnå en større målgruppe er der blandt andet valgt en anden skrift, da mange unge hurtigt bliver fanget af en sådan skrifttype end hvis det var Times New Roman. Dog er det



ikke alle computer der har denne skrifttype installeret, derfor har vi tilføjet et style, som referer til mappen hvor skrifttypen befinder sig. Denne style kan ses nedenstående:

```
1. <style>
2.     @font-face{
3.         font-family:"burbank";
4.         src:url("fonts/BurbankBigCondensed-Bold.otf")
5.     }
6.
7.     h1{font-family: burbank;}
8. </style>
9.
```

hvor font-family:"burbank" er skrifttypens navn,  
src:url("fonts/BurbankBigCondensed-Bold.otf") fil destinationen i mappen.

### 3.4.5. Test

Nedenstående ses en test af vores hjemmeside, som fungerer som det skal. Det hexadecimale "DAD" bliver konverteret til "3501", hvilket er korrekt. Dermed kan vi konkludere, at produktet fungerer.

Binary Number	Decimal Number	Hex Number
0	0	0
1	1	1
10	2	2
11	3	3
100	4	4
101	5	5
110	6	6
111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F
10000	16	10
10001	17	11
10010	18	12
10011	19	13
10100	20	14
10101	21	15
10110	22	16
10111	23	17
11000	24	18
11001	25	19

## 4. Resultatopgørelse

### 4.1. Indledende afsnit

I følgende afsnit opsummeres de områder vi har opnået i projektet, blandt andet i forhold til problemformuleringen og løsningsforslaget. Der sættes fokus på hvad vi har opnået i forhold til problemformuleringen og løsningsforslaget.

### 4.2. Hvad er opnået?

#### 4.2.1. Problemformulering

På baggrund af diverse problemstillinger og diskussioner er der indkredset et problem tilhørt med en problemformulering. Problemformuleringen har til formål, at oversætte vores overordnede emne til en konkret opgave, som vi gruppen har undersøgt og bearbejdet.

Nedenstående ses vores problemformulering, som har fokus på tal konvertering på et website.

I mange digitale systemer angives en fejlkode med et hexadecimalt tal, disse kan være uoverskuelige både at forstå og regne på. Kan gruppen fremstille et læringsprodukt, der kan konvertere disse tal til decimaltal fra 10-talssystemet?

Der kan konkluderes, at problemformuleringen er opfyldt således, at der i gruppen både er udviklet et konverteringsprogram samt en website, som giver brugeren en forklaring på det hexadecimale-, binære- og decimale talsystem. En dybere konklusion vil forekomme i næste to afsnits.

## 5. Evaluerende tanker

I starten af projektet blev vi ret hurtigt enige om at det, eller de, programmer vi skulle benytte til projektet var HTML, CSS og JavaScript. Kombinationen af disse programmer giver nemlig præcis samme grad af interaktivitet som Python, i virkeligheden højere, men også muligheden for design og æstetik som kan være svær at fremstille i Python.

Det faktum, at vi benytter os af den programkombination har på intet tidspunkt været en ulempe for os, tværtimod. Ligesom Python giver god feedback i fejlkoder gør konsollen i Google Chrome det også. Man meget let i koden kan referere til objekter og identiteter fra henholdsvis HTML og JavaScript og tilbage igen grundet deres nærmest symbiotiske opbygning.

Valget stod imellem HTML + JavaScript eller Python.

Med Python skulle vi naturligvis importere et modul såsom TkInter for at lave brugerinterface i stedet for at koden skulle køre udelukkende i terminalen. Det ville kræve at brugeren også installerede Python for at køre programmet - eller at man selv benytter

moduler såsom py2exe for at konvertere koden til en executable fil. Det er der ikke brug for i vores produkt da alle moderne operativsystemer som standard har minimum en webbrowser tilgængelig - de understøtter alle HTML og JavaScript.

Det skal nævnes at der i både Python og JavaScript er en enkelt måde at omskrive et hexadecimalt eller binært tal til decimal. I JavaScript hedder det `parseInt(x, y)` hvor man kan skrive `parseInt("101", 2)` og resultat vil blive 5. x er det/de tal der skal konverteres og y er basen i det talsystem der skal konverteres fra. I og med denne funktion allerede eksisterer, er vores program egentlig spild af arbejde og så alligevel ikke. Det, som funktionen benytter, er netop koden i vores program. Vi har altså egentlig opbygget vores egen `parseInt()`.

I Python er det nærmest den samme historie der bliver fortalt.

Kalder man sit hexadecimalt for s og denne er en string kan man benytte `i = int(s,16)` og `str(i)`. De omkring 50 linjer kode vi har skrevet i anden iteration er altså ganske unødvendige og kan erstattes af en enkelt `parseInt(s, 16)` i JavaScript eller `int(s, 16)` i Python. Dog har vi med projektet selv fået indsigt i hvordan det hexadecimalt foruden binære talsystem fungerer.

## 6. Konklusion

I dette projekt satte vi ud for at løse problemstillingen omkring, at det kan være et problem at forstå fejlkode i digitale systemer. For at forstå kunne slå disse fejlkode op skal man nemlig kunne løse hexadecimalt tal. Det kan være svært eller uoverskueligt at læse meget store tal, og til dette fremstillede vi vores hexadecimalt lommeregner, som til slut også fik binære til decimal konvertering. I og med der allerede findes værktøjer i både JavaScript såvel som Python til at konvertere tal (henholdsvis `parseInt(x,16)` og `int(x,16)`), har vi blot udviklet vores egen version af denne. Ikke nok med det får modtageren også en forklaring på talsystemer, både decimal, hexadecimal, binær og altså talsystemer generelt. Via vores produkt kan brugeren forhåbentlig opnå højere forståelse af hexadecimalt tal og anvende disse til noget.

## 7. Kildeliste

Websites			
Reference	Ophav	Link	Set
JavaScript Arrays	Codingame	<a href="https://www.codingame.com/playgrounds/">https://www.codingame.com/playgrounds/</a>	4. maj 2019

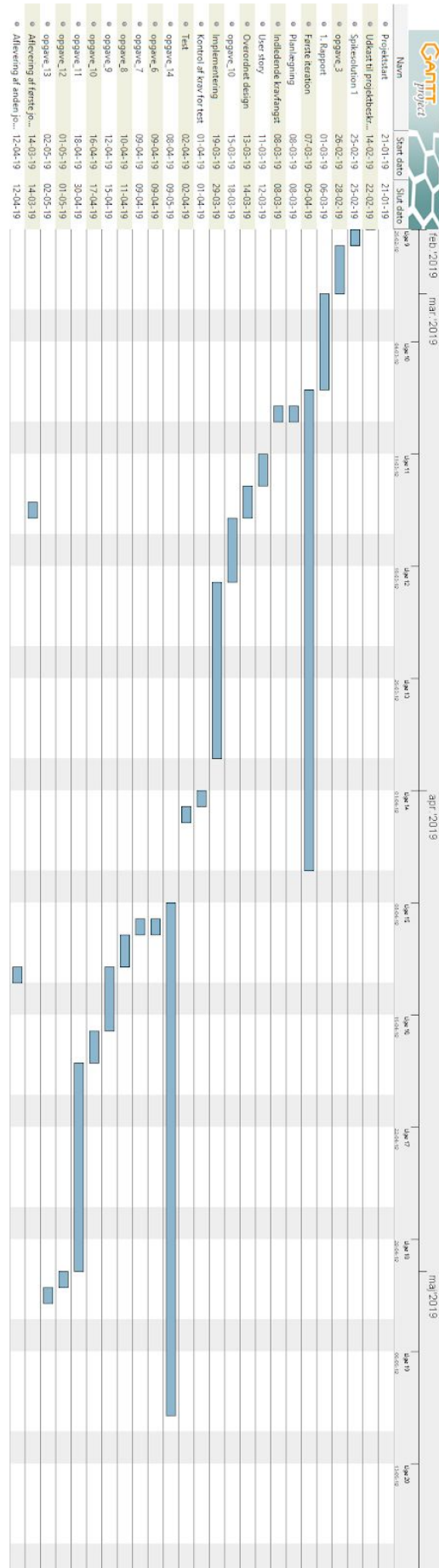
		<a href="#">6181/javascript-arrays---tips-tricks-and-examples</a>	
W3	W3	<a href="https://www.w3.org/html/">https://www.w3.org/html/</a>	1. maj 2019
CSS	CSS-tricks	<a href="https://css-tricks.com/">https://css-tricks.com/</a>	2. maj 2019
GitHub	GitHub	<a href="https://github.com/">https://github.com/</a>	29. marts 2019
Hexadecimale	Stackoverflow	<a href="https://stackoverflow.com/questions/57803/how-to-convert-decimal-to-hexadecimal-in-javascript">https://stackoverflow.com/questions/57803/how-to-convert-decimal-to-hexadecimal-in-javascript</a>	29. marts 2019

Websites		
Bog	Ophav	Set
HTML&CSS	Jon Duckett	6. marts 2019
JavaScript&JQuery	Jon Duckett	25. marts 2019

## 8. Bilag

### 8.1. Tidsplanlægning

Den fulde tidsplanlægning er vedhæftet herunder. Den har muliggjort, at vi ikke er blevet tidspresset med projektet, selv med andre sideløbende eksamensprojekter samt at den har sikret ordentlig kvalitet af de udviklede iteration af produktet



## 8.2. Kildekode

### 8.2.1. Første iteration

#### HTML:

```
1. <!DOCTYPE html>
2. <html>
3.
4.     <head>
5.         <meta charset="utf-8">
6.
7.         <title>Konvertering | Tal</title>
8.
9.         <link rel="stylesheet" href="main.css">
10.
11.        <script src="website.js"></script>
12.    </head>
13.
14.    <body>
15. <input type="text" id="indput" onkeyup="yeet()">
16. <button onclick="yeet()"> Klik her for konvertering af heksadecimal til
    decimal</button>
17. <p id="text-box"> Dit input: </p>
18. <p id="result"> Resultat: </p>
19.
20.
21.    </body>
22.
23. </html>
```

#### JavaScript:

```
1. console.log("Hello there!")
2. for (x=0; x < 10 ; x++){
3.     console.log(x);
4.
5. }
6.
7. function yeet(){
8.     conversion();
9.     document.getElementById("text-box").innerHTML = (document.getElementById('indput').value);
10. }
11.
12. function conversion(){
13.     var inputValue = document.getElementById("indput");
14.     if (inputValue == ("a")){
15.         var inputValue = (10);
16.         console.log(10);
17.     }
18.     else{
19.         console.log(11);
20.     }
21. }
```

## 8.2.2. Anden iteration

### JavaScript:

```
1. console.log("Hello there!");
2.
3. convert = true;
4.
5. function conversion(inputValue){
6.     var short = document.getElementById('result');
7.
8.     console.log(inputValue);
9.
10.    var mapping = {
11.        "0": "0",
12.        "1": "1",
13.        "2": "2",
14.        "3": "3",
15.        "4": "4",
16.        "5": "5",
17.        "6": "6",
18.        "7": "7",
19.        "8": "8",
20.        "9": "9",
21.        "A": "10",
22.        "B": "11",
23.        "C": "12",
24.        "D": "13",
25.        "E": "14",
26.        "F": "15"
27.    };
28.
29.    var srcString = inputValue.toString();
30.    var returnNum = 0;
31.    var i;
32.
33.    for (i = 0; i < srcString.length; i++) {
34.        returnNum += mapping[srcString[i]] * (16 ** (srcString.length - 1 - i));
35.    }
36.    console.log(returnNum);
37.    short.innerHTML = "Resultat: " + returnNum;
38. };
39.
40. function yeet(){
41.     var inputValue = document.getElementById('indput').value;
42.     document.getElementById("text-box").innerHTML = "Dit input var:
43.     "+(document.getElementById('indput').value);
44.     if (convert == true){
45.         conversion(inputValue);
46.         console.log("true");
47.     }
48.     else {
49.         console.log("stop");
```

```
49. }  
50. }
```

### 8.2.3. Tredje iteration

#### JavaScript:

```
1. // Konverteringsfunktionen der tager inputValue som parameter køres.  
2. function conversion(inputValue){  
3.     //variablen short referer til resultat  
4.     var short = document.getElementById('result');  
5.  
6.     var mapping = {  
7.         "0" : "0",  
8.         "1" : "1",  
9.         "2" : "2",  
10.        "3" : "3",  
11.        "4" : "4",  
12.        "5" : "5",  
13.        "6" : "6",  
14.        "7" : "7",  
15.        "8" : "8",  
16.        "9" : "9",  
17.        "A" : "10",  
18.        "B" : "11",  
19.        "C" : "12",  
20.        "D" : "13",  
21.        "E" : "14",  
22.        "F" : "15"  
23.     };  
24.  
25.     var srcString = inputValue.toString();  
26.     var returnNum = 0;  
27.     var i;  
28.  
29.     for (i = 0; i < srcString.length; i++) {  
30.         returnNum += mapping[srcString[i]] * (16 ** (srcString.length - 1 - i));  
31.     }  
32.     console.log(returnNum);  
33.     short.innerHTML = "Resultat: " + returnNum;  
34. };  
35.  
36. function yeet(){  
37.     var inputValue = document.getElementById('input').value;  
38.     document.getElementById("text-box").innerHTML = "Dit input var: " +  
        (document.getElementById('input').value);  
39.     if (convert == true){  
40.         conversion(inputValue);  
41.         console.log("true");  
42.     }  
43.     else {  
44.         console.log("stop");  
45.     }  
46. }
```



```
47.
48. // Binær --> Decimal konvertering starter her
49.
50.
51. // Funktionen der definerer mapping over binære værdier, foretager konvertering og printer resultatet starter her.
52. function binConversion(binValue){
53. // binShort refererer til den tekstboks i HTML-dokumentet der skal vise resultatet
54. var binShort = document.getElementById('dec-result');
55. // Her defineres defineres decimale værdier for hver binær
56. var binMapping = {
57.   "0" : "0",
58.   "1" : "1",
59. };
60. // Variabler defineres og den binære værdi bliver lavet om til string.
61. var binString = binValue.toString();
62. var binReturnNum = 0;
63. var i;
64. // For loop der tager i og længden af binString som parameter.
65. for (i = 0; i < binString.length; i++) {
66.   // Den matematiske beregning af det binære talsystem til decimal for i.
67.   binReturnNum += binMapping[binString[i]] * (2 ** (binString.length - 1 - i));
68. }
69. // Resultat printes
70. binShort.innerHTML = "Resultat: " + binReturnNum;
71. };
72. // Funktionen der eksekveres når der trykkes på konverteingskanppen på siden.
73. function binToDec(){
74. // Input gemmes som binValue
75. var binValue = document.getElementById('binær').value;
76. // Brugeren bliver oplyst om sit input
77. document.getElementById("bin-input").innerHTML = "Dit input var: " +
  (document.getElementById('binær').value);
78.   binConversion(binValue);
79. }
```

## HTML:

```
1. <!DOCTYPE html>
2. <html>
3.
4.   <head>
5.     <meta charset="utf-8">
6.
7.     <title>Konvertering | Tal</title>
8.
9.     <link rel="stylesheet" href="main.css">
10.
11.    <script src="website.js"></script>
12.
13.    <style>
14.      @font-face{
15.        font-family:"burbank";
16.        src:url("fonts/BurbankBigCondensed-Bold.otf")
17.      }
```

```
18.
19.         h1{font-family: burbank;}
20.     </style>
21.
22. </head>
23.
24. <body>
25.
26.     <div class="top_bar">
27.         <h1 style="font-family:burbank big condensed">Tal Konvertering</h1>
28.     </div>
29.
30.     <input type="text" id="indput" onkeyup="yeet()" placeholder="Skriv venligt">
31.
32.     <button onclick="yeet()" class="klik" style="font-family:burbank big
condensed">Konverter</button>
33.
34.     <p class="text-box" id="text-box" style="font-family:burbank big condensed">
Dit input: </p>
35.
36.     <p class="resultat" id="result" style="font-family:burbank big condensed">
Dit resultat: </p>
37.
38.     <h3 class="forklaring_hex">Hvad er hexadecimal talsystem?</h3>
39.
40.     <p class="forklaring_hex1">I det talsystem vi til dagligdag
benytter, har vi basen 10. <br> Det betyder altså, at når man
41.         når over 9, begynder man <br> forfra ved det første tal og sætter
et 0 bag på. Så skriver vi <br> altså 10.
42.         Det gør man ikke i det hexadecimale talsystem.
43.         <br> I dette system har man i stedet basen, så man skriver tallene
<br> fra 0 og op til og med 15, eller
44.         det hedder i det hexadecimale <br> talsystem 'f'. I dette system
skriver man ikke 10, man det <br> første
45.         bogstav i alfabetet og det næste indtil man når 'f' - eller <br>
15 i decimalsystemet/10-talssystemet.
46.         Herunder ses <br> sammenligningen imellem de to, så man kan danne
sig et lille <br> overblik på hvordan de to
47.         systemer hænger sammen.
48.         For at <br> omregne fra hexadecimal til decimal, læser man fra
højre, <br> ganger man symbolets værdi
49.         med 16 opløftet i symbolets <br> position fra højre side. Det vil
sige, står der f.eks. 1A hedder <br>
50.         udregningen:  $10 \cdot 16^1 + 1 \cdot 16^0 = 26$ . <br> <br> Et skema over denne
konvertering kan ses på højre side.
51.     </p>
52.
53.     <div class="con_left">
54.         <button class="convert" style="font-family:burbank big
condensed">Hexadecimale Talsystem</button>
55.     </div>
56.
57.     <div class="con_right">
```

```

58.         <button class="convert" style="font-family:burbank big condensed">Binære
Talsystem</button>
59.     </div>
60.
61.     <input id="binær" type="text1" placeholder="Skriv venligt">
62.
63.     <button onclick="binToDec()" class="klik1" style="font-family:burbank big
condensed">Konverter</button>
64.
65.     <p class="bin-input" id="bin-input" style="font-family:burbank big
condensed"> Dit input: </p>
66.
67.     <p class="dec-result" id="dec-result" style="font-family:burbank big
condensed"> Dit resultat: </p>
68.     <table class="ntable" border="1" style="text-align:center">
69.
70.     <h3 class="forklaring_bin">Hvad er binær talsystem?</h3>
71.
72.     <p class="forklaring_bin1">Det binære talsystem, eller også kaldt
totalssystemet som <br> kun består af to cifre: 1 og 0. Det binære talsystem bruges
<br> blandt andet ved lagring af data på medier som hukort/- <br> strimmel,
magnetbånd, eksempelvis DAT, magnetstribes på <br> ID-kort, DVD og harddisk.
Princippets opbygning er ens <br> med titalssystemet, idet der kan være ét ciffer
fra 0-9 på <br> hver plads, men i det binære talsystem kan der kun og netop <br> kun
være ét af to ciffer, herunder 0 eller 1. I det binære <br> talsystem læses der fra
højre mod venstre, hvilket vil sige <br> hvis der skal skrives 10, så er det binære
tal lig med 1010, <br> eller 14, så er det 1110, 15 er 1111 osv.
73.     </p>
74.     <br> <br> <br>
75.     <thead class="table1">
76.     <tr>
77.         <th>Binary<br>Number</th>
78.         <th>Decimal<br>Number</th>
79.         <th>Hex<br>Number</th>
80.     </tr>
81.     </thead>
82.     <tbody><tr>
83.         <td>0</td>
84.         <td>0</td>
85.         <td>0</td>
86.     </tr>
87.     <tr>
88.         <td>1</td>
89.         <td>1</td>
90.         <td>1</td>
91.     </tr>
92.     <tr>
93.         <td>10</td>
94.         <td>2</td>
95.         <td>2</td>
96.     </tr>
97.     <tr>
98.         <td>11</td>

```

```
99.         <td>3</td>
100.        <td>3</td>
101.    </tr>
102.    <tr>
103.        <td>100</td>
104.        <td>4</td>
105.        <td>4</td>
106.    </tr>
107.    <tr>
108.        <td>101</td>
109.        <td>5</td>
110.        <td>5</td>
111.    </tr>
112.    <tr>
113.        <td>110</td>
114.        <td>6</td>
115.        <td>6</td>
116.    </tr>
117.    <tr>
118.        <td>111</td>
119.        <td>7</td>
120.        <td>7</td>
121.    </tr>
122.    <tr>
123.        <td>1000</td>
124.        <td>8</td>
125.        <td>8</td>
126.    </tr>
127.    <tr>
128.        <td>1001</td>
129.        <td>9</td>
130.        <td>9</td>
131.    </tr>
132.    <tr>
133.        <td>1010</td>
134.        <td>10</td>
135.        <td>A</td>
136.    </tr>
137.    <tr>
138.        <td>1011</td>
139.        <td>11</td>
140.        <td>B</td>
141.    </tr>
142.    <tr>
143.        <td>1100</td>
144.        <td>12</td>
145.        <td>C</td>
146.    </tr>
147.    <tr>
148.        <td>1101</td>
149.        <td>13</td>
150.        <td>D</td>
151.    </tr>
```

```
152.         <tr>
153.             <td>1110</td>
154.             <td>14</td>
155.             <td>E</td>
156.         </tr>
157.         <tr>
158.             <td>1111</td>
159.             <td>15</td>
160.             <td>F</td>
161.         </tr>
162.         <tr>
163.             <td>10000</td>
164.             <td>16</td>
165.             <td>10</td>
166.     </tr>
167.     <tr>
168.         <td>10001</td>
169.         <td>17</td>
170.         <td>11</td>
171.     </tr>
172.     <tr>
173.         <td>10010</td>
174.         <td>18</td>
175.         <td>12</td>
176.     </tr>
177.     <tr>
178.         <td>10011</td>
179.         <td>19</td>
180.         <td>13</td>
181.     </tr>
182.     <tr>
183.         <td>10100</td>
184.         <td>20</td>
185.         <td>14</td>
186.     </tr>
187.     <tr>
188.         <td>10101</td>
189.         <td>21</td>
190.         <td>15</td>
191.     </tr>
192.     <tr>
193.         <td>10110</td>
194.         <td>22</td>
195.         <td>16</td>
196.     </tr>
197.     <tr>
198.         <td>10111</td>
199.         <td>23</td>
200.         <td>17</td>
201.     </tr>
202.     <tr>
203.         <td>11000</td>
204.         <td>24</td>
```

```

205.         <td>18</td>
206.     </tr>
207.     <tr>
208.         <td>11001</td>
209.         <td>25</td>
210.         <td>19</td>
211.     </tr>
212.     <tr>
213.         <td>11010</td>
214.         <td>26</td>
215.         <td>1A</td>
216.     </tr>
217.     <tr>
218.         <td>11011</td>
219.         <td>27</td>
220.         <td>1B</td>
221.     </tr>
222.     <tr>
223.         <td>11100</td>
224.         <td>28</td>
225.         <td>1C</td>
226.     </tr>
227.     <tr>
228.         <td>11101</td>
229.         <td>29</td>
230.         <td>1D</td>
231.     </tr>
232.     <tr>
233.         <td>11110</td>
234.         <td>30</td>
235.         <td>1E</td>
236.     </tr>
237.     <tr>
238.         <td>11111</td>
239.         <td>31</td>
240.         <td>1F</td>
241.     </tr>
242.     <tr>
243.         <td>100000</td>
244.         <td>32</td>
245.         <td>20</td>
246.     </tr>
247.     <tr>
248.         <td>1000000</td>
249.         <td>64</td>
250.         <td>40</td>
251.     </tr>
252.     <tr>
253.         <td>10000000</td>
254.         <td>128</td>
255.         <td>80</td>
256.     </tr>
257.     <tr>

```

```
258.         <td>100000000</td>
259.         <td>256</td>
260.         <td>100</td>
261.     </tr>
262. </tbody></table>
263.
264. </body>
265.
266. </html>
```

## CSS:

```
1.  * {
2.      margin: 0;
3.      padding: 0;
4.  }
5.
6.  body {
7.      background-color: white;
8.  }
9.
10. .top_bar {
11.     width: auto;
12.     height: 120px;
13.     background-color: orange;
14.     color: black;
15. }
16.
17. h1 {
18.     color: white;
19.     text-align: center;
20.     font-size: 100px;
21.     padding: 10px;
22. }
23.
24. input[type=text]{
25.     width: 17%;
26.     border: 2px solid #aaa;
27.     border-radius: 4px;
28.     margin: 8px 0;
29.     outline: none;
30.     padding: 8px;
31.     box-sizing: border-box;
32.     transition: .3s;
33.     position: absolute;
34.     top: 33%;
35.     left: 13.9%;
36. }
37.
38. input[type=text]:focus{
39.     border-color: dodgerBlue;
40.     box-shadow: 0 0 8px 0 dodgerBlue;
41. }
42.
```

```
43. .klik {
44.     position: absolute;
45.     top: 34%;
46.     left: 31.3%;
47.     width: 7.3%;
48.     height: 4%;
49.     border: 2px solid #aaa;
50.     border-radius: 4px;
51.     color: orange;
52.     font-size: 25px;
53. }
54.
55. .klik1 {
56.     position: absolute;
57.     top: 34%;
58.     right: 14%;
59.     width: 7.3%;
60.     height: 4%;
61.     border: 2px solid #aaa;
62.     border-radius: 4px;
63.     color: orange;
64.     font-size: 25px;
65. }
66.
67. .text-box {
68.     position: absolute;
69.     top: 43%;
70.     left: 13.8%;
71.     font-size: 30px;
72. }
73.
74. .resultat {
75.     position: absolute;
76.     top: 50%;
77.     left: 13.8%;
78.     font-size: 30px;
79. }
80.
81. .forklaring_hex {
82.     position: absolute;
83.     top: 65%;
84.     left: 13.8%;
85. }
86.
87. .forklaring_hex1 {
88.     position: absolute;
89.     top: 69%;
90.     left: 13.8%;
91. }
92.
93. .ntable {
94.     border: 1px;
95.     margin-left: auto;
```



```
96.     margin-right:auto;
97. }
98.
99. .forklaring_bin {
100.     position: absolute;
101.     top: 65%;
102.     right: 24.2%;
103. }
104.
105. .forklaring_bin1 {
106.     position: absolute;
107.     top: 69%;
108.     left: 61.5%;
109. }
110.
111. .bin-input {
112.     position: absolute;
113.     top: 43%;
114.     right: 32.5%;
115.     font-size: 30px;
116. }
117.
118. .dec-result {
119.     position: absolute;
120.     top: 50%;
121.     right: 30.7%;
122.     font-size: 30px;
123. }
124.
125. input[type=text1]{
126.     width:17%;
127.     border:2px solid #aaa;
128.     border-radius:4px;
129.     margin:8px 0;
130.     outline:none;
131.     padding:8px;
132.     box-sizing:border-box;
133.     transition:.3s;
134.     position: absolute;
135.     top: 33%;
136.     right: 21.5%;
137. }
138.
139. input[type=text1]:focus{
140.     border-color:dodgerBlue;
141.     box-shadow:0 0 8px 0 dodgerBlue;
142. }
143.
144. .con_left {
145.     width: 24.5%;
146.     position: absolute;
147.     top: 21%;
148.     left: 14%;
```

```
149.     margin: auto;
150.     text-align: center;
151.     border: 2px solid #ffa500;
152. }
153.
154. img {
155.     display: block;
156.     margin-left: auto;
157.     margin-right: auto;
158.     margin-top: 195px;
159. }
160.
161. .convert {
162.     background-color: white;
163.     color: black;
164.     padding: 16px;
165.     font-size: 40px;
166.     border: none;
167. }
168.
169. .con_right {
170.     width: 24.5%;
171.     position: absolute;
172.     top: 21%;
173.     right: 14%;
174.     margin: auto;
175.     text-align: center;
176.     border: 2px solid #ffa500;
177. }
178.
179. .select-style {
180.     border: 1px solid #ccc;
181.     width: 120px;
182.     border-radius: 3px;
183.     overflow: hidden;
184.     background: #fafafa url("img/icon-select.png") no-repeat 90% 50%;
185. }
186.
187. .select-style select {
188.     padding: 5px 8px;
189.     width: 130%;
190.     border: none;
191.     box-shadow: none;
192.     background: transparent;
193.     background-image: none;
194.     -webkit-appearance: none;
195. }
196.
197. .select-style select:focus {
198.     outline: none;
199. }
200.
```

## 8.3. Licenser

### 8.3.1. Brackets

Brackets er udgivet under MIT-licensen som er vedhæftet herunder:

Copyright (c) 2012 - present Adobe Systems Incorporated. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

### 8.3.2. Atom

Atom er også open-source og kan hentes via deres hjemmeside <https://atom.io/> eller via deres repository på GitHub <https://github.com/atom/atom>

Copyright (c) 2011-2018 GitHub Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

### 8.3.3. Licens - produkt

Vores produkt er udviklet under GNU Public License v3.0, således at alle frit må bruge koden til både private såvel som kommercielle projekter. De må foretage alle de ændringer de har lyst til, og må gerne distribuere koden selv.

Dog er de selv ansvarlige for hvad der sker med koden derfra. Vi bærer intet ansvar for hvad der videre sker med programmet. Licensen kan findes her<sup>11</sup>:

[https://github.com/NathiNugget/Eksamensprojekt\\_konverteringer/blob/master/LICENSE](https://github.com/NathiNugget/Eksamensprojekt_konverteringer/blob/master/LICENSE)

---

<sup>11</sup> [https://github.com/NathiNugget/Eksamensprojekt\\_konverteringer/blob/master/LICENSE](https://github.com/NathiNugget/Eksamensprojekt_konverteringer/blob/master/LICENSE)