

Internets uppbyggnad



Internet - decentralisert

- Utvecklat av militären
- Om en stad wipas ut måste informationen fortfarande kunna komma fram



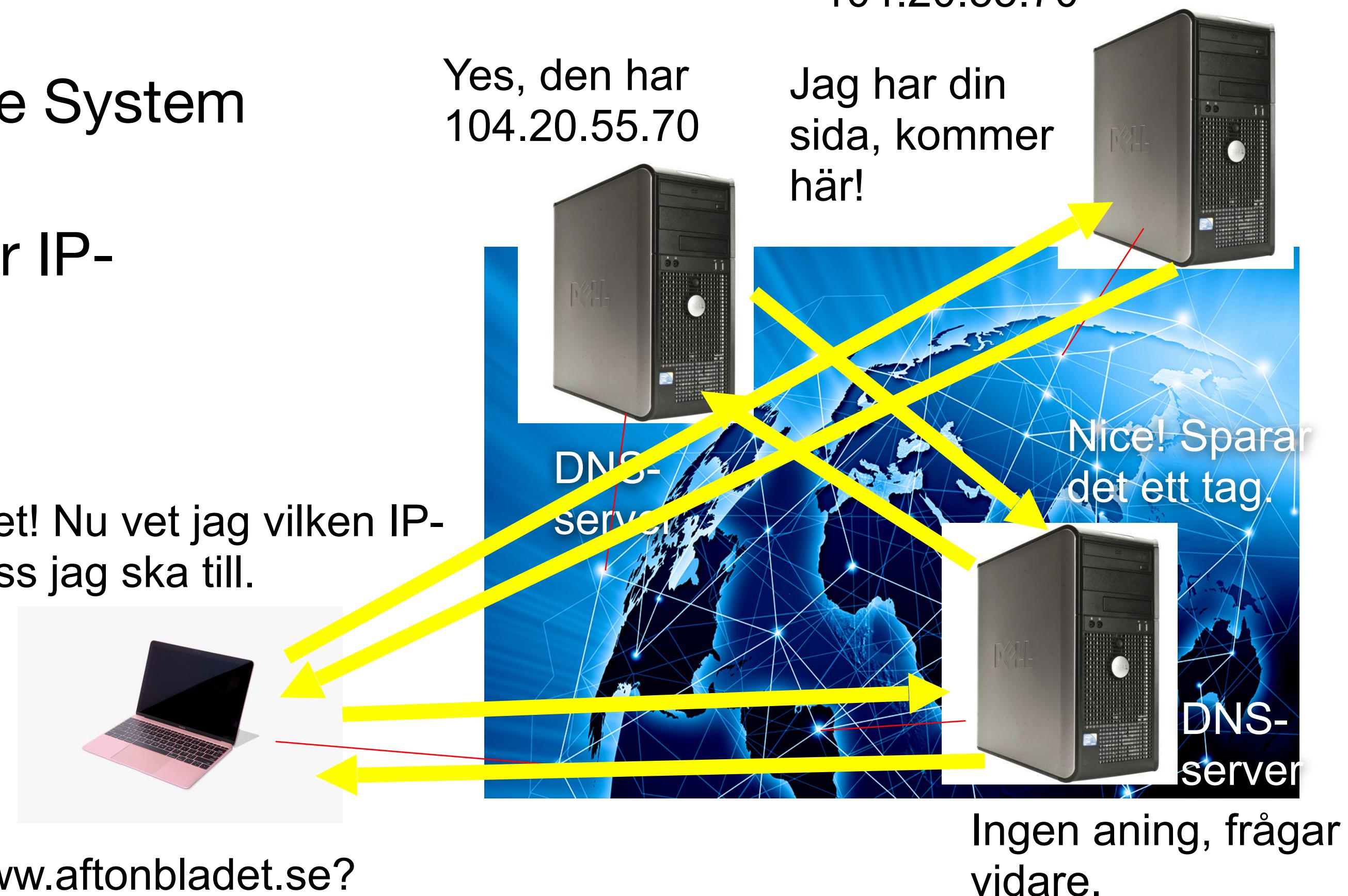
Internet - IP-adress

112.39.10.114 2001:0db8:0000:0000:0000:00
 00:1428:07ab/64



Internet - DNS

- Domain Name System
- Adressbok för IP-adresser

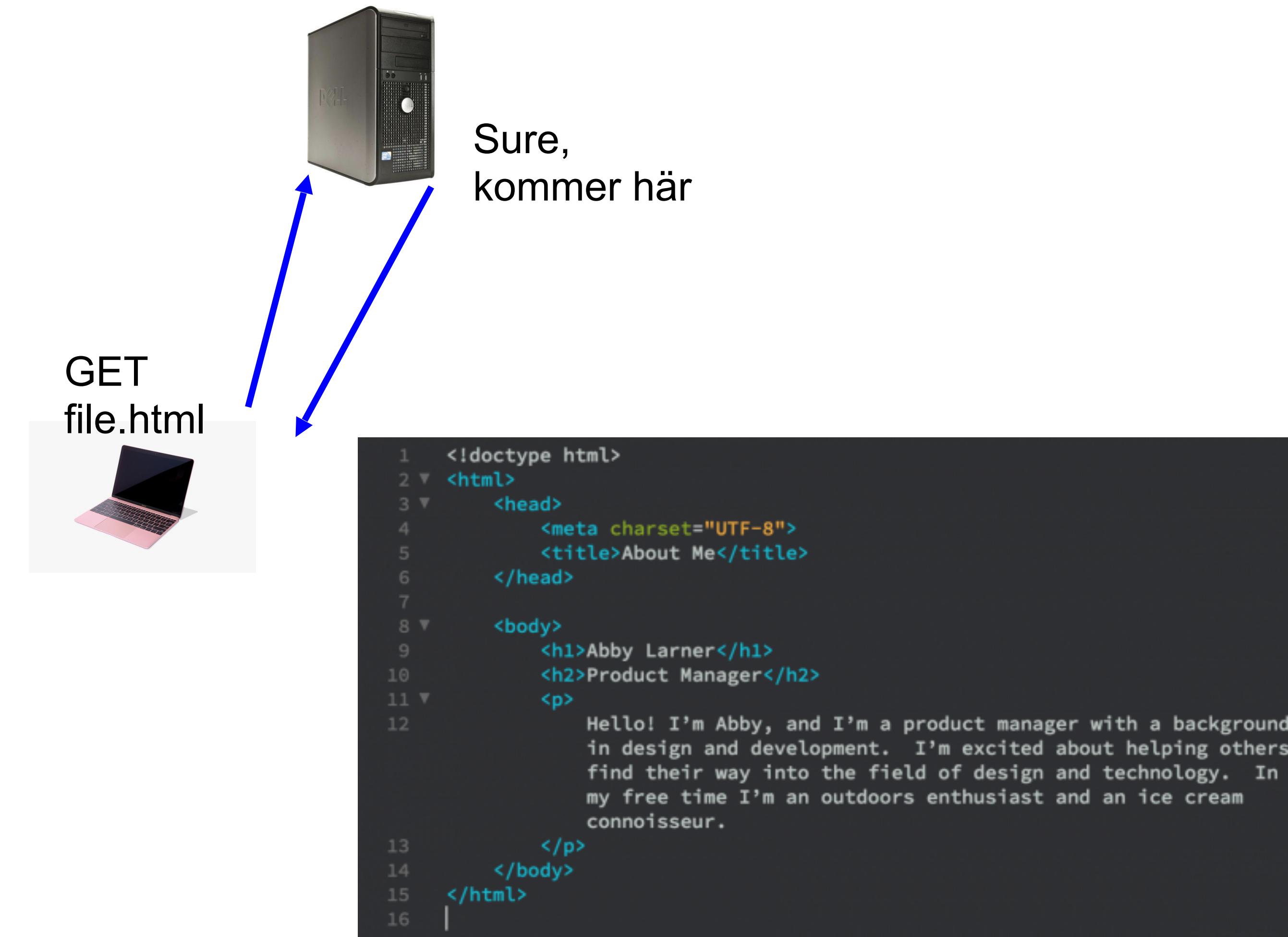


Internet - http - vanliga statusar

- 200: OK
 - 301: Moved Permanently
 - 401: Unauthorized
 - 403: Forbidden
 - 404: Not Found
 - 500: Internal Server Error
- HyperText Transfer Protocol
 - används för att överföra webbsidor
 - definierar åtta kommandon
 - GET
 - HEAD
 - POST
 - PUT
 - DELETE
 - TRACE
 - OPTIONS
 - CONNECT
 - Svaret från webbservern innehåller en HTTP-statuskod

HTML - formulär

- Servern kan skicka information till besökaren (klienten) i form av html, css osv.



HTML - formulär

- Hur kan besökaren (klienten) skicka information till servern?
- Varför skulle vi vilja att besökaren kan skicka information till oss?
- Hur kan vi hantera informationen som besökaren skickar till oss? Kan vi göra det med html?
 - Nej, vi behöver ett programmeringsspråk som kan tolka vad vi skickat upp; PHP, C#, Java



HTML - formulär

- Saker vi kan göra med information besökaren skickar:
 - Posta ett inlägg a la facebook
 - Visa en kategori av böcker som besökaren valt
 - Autentisera användare
 - Mer?

HTML - formulär

- Säkerhet (för framtiden)
 - Lita aldrig på något användaren skickar.
 - Rensa/säkra!

Vad är ett webbformulär?

- Ett sätt att skicka och hämta information
- Ett eller flera fält för inmatning av data
 - Text
 - Siffror
 - Datum
 - Dropdowns
 - Kryssrutor
 - Radioknappar

Hur använder man webbformulär?

- Formulär består av ett eller flera inputfält inuti en FORM-tagg
- Innehåller ofta en submit-knapp som skickar iväg informationen i formuläret

```
1 <form>  
2   Namn:<br>  
3   <input type="text" name="name1"><br>  
4   Meddelande:<br>  
5   <textarea name="message"></textarea><br>  
6   <input type="submit">  
7 </form>
```

Namn:

Meddelande:

Skicka

FORM-taggen

```
1 <form method="POST"  
2     action="send.php">  
3     <label for="name">Name</label>  
4     <input id="name" type="text">  
5     <button type="submit">Skicka</button>  
6 </form>
```

- FORM-taggen representerar hela formuläret.
- Method-attributet beskriver hur formuläret ska skickas.
- Action-attributet beskriver vad som ska hänta när man skickar in formuläret.
- Dessa är bara relevanta när man håller på med backend-programmering.

FORM-taggen

```
1 <form method='POST' ←  
2     action="send.php">  
3 <label for="name">Name</label>  
4 <input id="name" type="text">  
5 <button type="submit">Skicka</button>  
6 </form>
```

- Hur formuläret ska skickas

Input-taggen - label

```
1 <label for="name">Name</label>
2 <input id="name" type="text">
```

Name

- Input-taggen paras alltid med en label-tagg som beskriver syftet med input-fältet.
- Label-taggen har ett attribut for som kopplas till input-taggens id.
- Label används för att göra det lättare att nå fältet, framför allt till små komponenter typ radioknappar och checkboxar.

Input-taggen - type

```
1 <label for="name">Name</label>
2 <input id="name" type="text">
```

- Type-attributet beskriver vilken typ av data som ska matas in.
 - Text
 - Number
 - Checkbox
 - Radio
 - Password
 - Email
 - Url
 - Color
 - Date

Input-taggen - placeholder

- Placeholder-attributet sätter en hjälptext inuti text-fältet som försvinner när fältet får fokus.
- Placeholder-texter används ofta för att förtydliga för användaren vad de ska fylla i.

```
1 <label for="name">Name</label>
2 <input id="name" type="text" placeholder="Skriv ditt namn här">
```

Name

Input-taggen - value

```
1 <label for="name">Name</label>
2 <input id="name" type="text" value="Micke">
```

Name

- Value-attributet fyller inputfältet med en förifyllt värde som inte försvinner när fältet får fokus.
- Value-texten används för redan känt data som användaren kan välja att ändra eller inte.

Textarea-taggen

```
‐ <label for="message">Meddelande</label>
‐ <textarea id="message" maxlength="140">Value</textarea>
```

Meddelande

- Textarea är ett inputfält för längre textstycken.
- Attributet maxlength sätter en gräns på antal tecken i fältet.
- Textarea har inget value-attribut.

Select-taggen

```
1 <label for="colours">Välj färg</label>
2 <select id="colors">
3   <option value="blue" selected>Blå</option>
4   <option value="green">Grön</option>
5   <option value="red">Röd</option>
6 </select>
```

Välj färg 

- Select-taggen skapar en dropdown lista.
- Alternativen i listan beskrivs i option taggar med value-attribut.
- Selected i option-taggen sätter det alternativet som förvalt.

Disabled och Required

```
<label for="name">Name</label>
<input id="name" type="text" disabled>

<label for="email">Epost</label>
<input id="email" type="email" required>
```

Name Epost

- Button, inputfält, textarea och select-taggar kan låsas genom att sätta ett disabled-attribut på dem.
- Inputfält, textarea och select-taggar kan göras obligatoriska genom att sätta ett required -attribut på dem.

Bygg formulär

Bygg ett formulär med lämpliga komponenter med lämpliga id:n ni ber era besökare om information:

- Namn
- Lösenord
- Födelsedatum (datum-väljare)
- E-post
- Ålder
- Personlig presentation
- Favoritfärg (färg-väljare)
- Favoritveckodag (dropdown)
- Mac vs. PC (radio-buttons)
- TV-serier du sett (check-box)

HTML - formulär

```
<form method="post" action="some_page.php">
    Namn:<br>
    <input type="text" id="name" placeholder="Skriv ditt namn här"><br>

    Ålder:
    <select id="age">
        <option value="age_group1">0-30</option>
        <option value="age_group2">30-60</option>
        <option value="age_group3">60-</option>
    </select>
</form>
```

HTML - formulär

```
<form method="post" action="some_page.php">
    Intressen:<br>
    <input type="checkbox" name="hobbies" value="fotboll">Fotboll<br>
    <input type="checkbox" name="hobbies" value="hockey">Hockey<br>

    Favorit-färg:
    <input type="radio" name="color" value="yellow">Gul<br>
    <input type="radio" name="color" value="red">Röd<br>
</form>
```

HTML - formulär

```
<form method="post" action="some_page.php">
    <input type="number" min="0" max="100" step="2"><br>
    <input type="email"><br>
    <input type="color"><br>
</form>
```

Event

”DOM Events are sent to notify code of interesting things that have taken place. Each event is represented by an object which is based on the Event interface, and may have additional custom fields and/or functions used to get additional information about what happened. Events can represent everything from basic user interactions to automated notifications of things happening in the rendering model.”

<https://developer.mozilla.org/en-US/docs/Web/Events>

Interaktion med HTML/CSS

- Nå HTML element från JavaScript-filen
- Vad är/gör dessa delar?
- VERSALER/gemener är viktigt i JS.

```
document.addEventListener(event, function (e) ) ;
```

Interaktion med HTML/CSS

- "Nå" HTML-element från JavaScript-filen
- getElementById ger oss en referens till objektet som representerar elementet med detta id i DOM:en.
- Låter oss hämta information från elementet eller ändra på det.

Fördefinierad sträng som visar vilket event vi lyssnar på

Anonym funktion

```
document.addEventListener("DOMContentLoaded", function(e) {  
    let headline = document.getElementById("headline");  
    let text = headline.innerHTML;  
});
```

Interaktion med HTML/CSS

- Ändra HTML element från JavaScript-filen

```
document.addEventListener("DOMContentLoaded", function(event) {  
    let headline = document.getElementById("headline");  
    let oldText = headline.innerHTML;  
    headline.innerHTML = "New Headline";  
    headline.style.color = "#0000ff";  
});
```

Interaktion med HTML/CSS

- Hämta information från inputfält
- Från input-komponenter hämtar vi värdet med `.value` istället för `.innerHTML`.
- Vi kan kolla om en checkbox är kryssad med hjälp av `.checked`.

```
document.addEventListener("DOMContentLoaded", function(event) {  
    let name = document.getElementById("name").value;  
    let acceptRules = document.getElementById("rules").checked;  
});
```

Andra event

- Vad har vi för event här?
- Varför är de wrappade?

```
document.addEventListener("DOMContentLoaded", function(event) {  
    let button = document.getElementById("btnSend");  
    button.addEventListener("click", function(event) {  
        alert("You clicked the button!");  
    }  
});
```

Event

- Vad är det för typ av event?

```
<button id="calc">Beräkna</button>
```

```
// - - -
```

```
// Spara beräkna-knappen i en variabel.  
let calc_button = document.getElementById("calc");
```

```
// Ha koll på om någon klickar på beräkna-knappen.  
calc_button.addEventListener("click", function(event) {  
    // Do something.  
});
```

Dags att testa själva!

- Utgå från ditt formulär och att användaren har klickat på en knapp. Om du ska skriva ut något, använd console.log.
 - Skriv ut "Hej namn!" där namn är användarens namn.
 - Skriv ut hur gammal användaren är och hur många dagar det är tills nästa födelsedag.
 - Skriv ut om den ålder användaren har angivit stämmer. (true/false)
 - Byt färg på sidan till den färg användaren har angivit.
 - Skriv ut Mac eller PC beroende på användarens val.
 - Skriv ut alla TV-serier användaren har kryssat i.
 - Lägg till ett div-element och skriv ut användarens presentation i den.

Yatzy

- Ge era inputs id:n, typ "player1_ones". Det räcker att göra för en spelare, vi kommer att ändra detta längre fram.
- Skapa en Beräkna-knapp.
- När man klickar på Beräkna-knappen ska summan av alla ettor, tvåor, treor, fyror, femmor och sexor beräknas och summan ska uppdateras. Det räcker att göra det för en spelare.
 - Tänk på att när man hämtar ett värde från en input, så har den datatypen String och måste konverteras till ett tal innan man kan räkna med värdet.
 - Vad händer om användaren t ex inte har några treor?
 - En del har gjort ett `input`-fält för sin summa, en del har en tom `td` eller liknande. Tänk på att skriva ut värdet på olika sätt, t ex `innerHTML` om ni har ett element typ `p`, `span`, `td` eller liknande, `value` om ni har en `input`.
- Om spelarens summa är minst 63 poäng ska användaren få 50 poäng i bonus.

Funktioner

- Vilken typ är parametrarna?

Sträng

Funktion

```
document.addEventListener ("DOMContentLoaded", function(event) {  
    let headline = document.getElementById("headline");  
  
    headline.innerHTML = "Lorem ipsum";  
}) ;
```

Funktioner

- Det måste betyda att vi kan ange en "extern" funktion som parameter!
- Lägg märke till att funktionen namn ska anges utan parentes.
- Vi vill inte köra funktionen, vi vill bara ange vilken funktion som ska köras när det är dags.

```
function something(event) {  
    let headline = document.getElementById("headline");  
  
    headline.innerHTML = "Lorem ipsum";  
}  
  
document.addEventListener ("DOMContentLoaded", something);
```

Funktioner

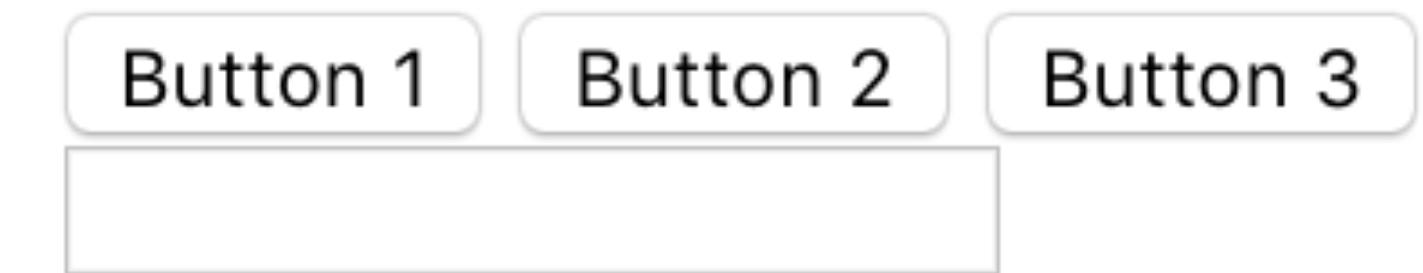
- *Uppgift:* Gör ett formulär med ett text-fält och tre knappar.
- När man klickar på någon av knapparna ska en funktion anropas som slumpar fram ett tal och uppdaterar värdet i textrutan med det.
- Det ska vara samma funktion som anropas vilken knapp man än trycker på.

```
let slump = Math.random();  
console.log(slump);
```



Target

- Om det är samma funktion som körs, kan vi ta reda på vilken knapp användaren tryckte på?



```
function something(event) {  
  console.log(event.target)  
}
```

```
document.addEventListener ("click", something);
```

Event bubbling

- Om användaren gör något som triggar ett event kommer browsern i första hand att kolla om det finns någon eventlyssnare för det aktuella elementet.
- Om det inte finns någon för det elementet kommer den att ”bubbla” uppåt, dvs att kolla i föräldrar i stigande led tills den eventuellt hittar någon.
- Om vi behöver veta vilket element som faktiskt orsakade eventet kan vi använda *target*.

Default

- Ibland vill man inte att det som normalt händer när ett event utlöses ska hända.

```
calc_button.addEventListener("click", function(event) {  
    event.preventDefault();  
    // Do something.  
});
```

Hindra att sidan laddas om

- Ibland när man gör ett formulär ser det ut som att inget händer när sidan egentligen laddas om.
- Ett sätt att förhindra att sidan laddas om är preventDefault().
- Ett annat är att göra en button med type button. Kan låta konstigt, men det funkar.

Andra event

- Vad har vi för event här?
- Varför är de wrappade?

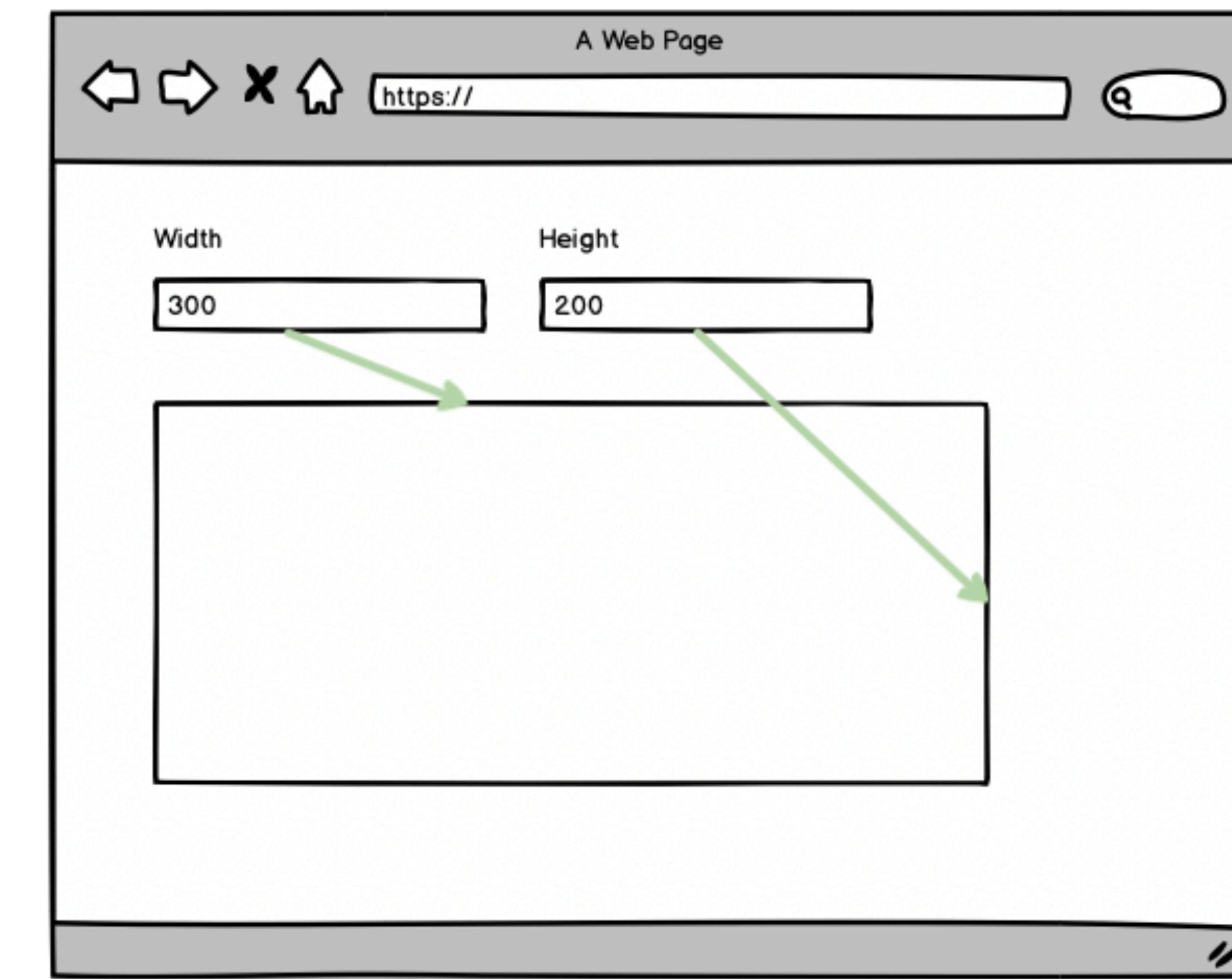
```
document.addEventListener("DOMContentLoaded", function(event) {  
    let button = document.getElementById("btnSend");  
    button.addEventListener("click", function(event) {  
        alert("You clicked the button!");  
    })  
}) ;
```

Andra event-typer

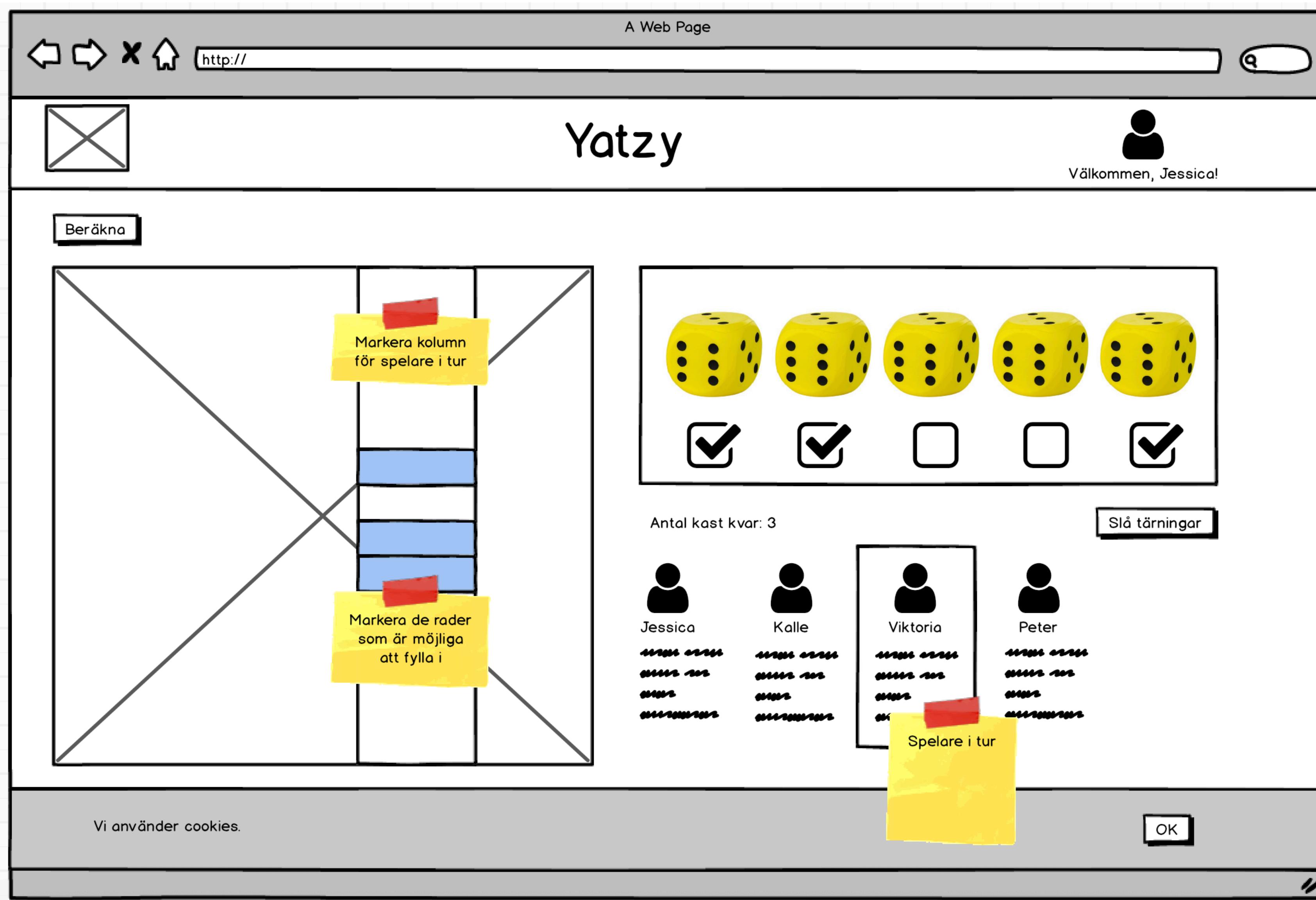
- keypress
- keydown
- keyup
- click
- mousedown
- mouseup
- change
- submit
- <https://developer.mozilla.org/en-US/docs/Web/Events>

Uppgifter: event

- Skapa en `<div>`-tagg med en synlig border.
- Med hjälp av en color-komponent, ge div:en en ny färg.
- Skapa input-fält som låter användaren bestämma hur bred och hög div-en ska vara. Kontrollera värdet så fort användaren har tryckt på en tangent.



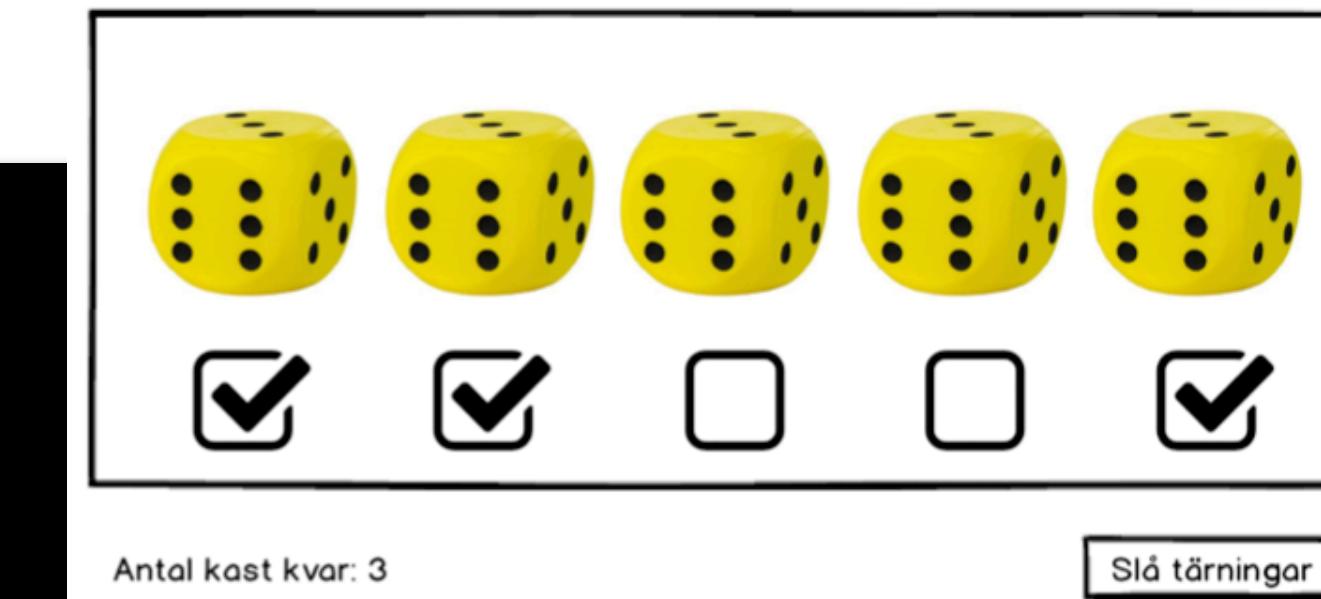
Yatzy - layout



Yatzy

- Ta bort beräkna-knappen och gör så att summan uppdateras när något av fälten (ettor, tvåor t o m sexor) ändras. Gör det bara för en spelare, ni kommer troligen att vilja lösa detta mer generellt senare.
- *Extra uppgift:* Gör ett tärningsformulär med fem text-rutor, fem kryssrutor och en knapp.
 - När man klickar på knappen ska alla rutorna få ett nytt tal (simulera tärningskast).
 - *Extra uppgift:* De tärningar som är "kryssade" ska inte få ett nytt tal.
 - *Extra uppgift:* Håll reda på hur många kast spelaren har kvar. (Tre från början.)
 - *Extra uppgift:* Fundera på hur man skulle kunna göra för att få till bilden nedan, att visa en tärning med rätt antal ögon istället för ett tal i en textruta.

```
var slump = Math.random();  
console.log(slump);
```



getElementById

- Returnerar elementet som matchar.

```
var element = document.getElementById(id);
```

getElementsByClassName

- Returnerar alla element som matchar.
- Returnerar en HTMLCollection som är rätt lik en array.

```
// Get all elements that have a class of 'test'  
let elements = document.getElementsByClassName('test')  
  
// Get all elements that have both the 'red' and 'test'  
// classes  
let elements = document.getElementsByClassName('red test')  
  
// Get all elements that have a class of 'test', inside of an  
// element that has the ID of 'main'  
let elements = document.getElementById('main').getElementsByClassName('test')
```

HTMLCollection

- The HTMLCollection interface represents a generic collection (array-like object similar to arguments) of elements (in document order) and offers methods and properties for selecting from the list.
- An HTMLCollection in the HTML DOM is live; it is automatically updated when the underlying document is changed.
- <https://developer.mozilla.org/en-US/docs/Web/API/HTMLCollection>

HTMLCollection

```
var list = document.getElementsByClassName("events");
for (let item of list) {
    console.log(item.id);
}
```

getElementsByName

- Returnerar alla element som matchar.

```
// Get all elements by the tag  
let elements = document.getElementsByTagName('p');
```

NodeList

- NodeList objects are collections of nodes, usually returned by properties such as Node.childNodes and methods such as document.querySelectorAll().
- <https://developer.mozilla.org/en-US/docs/Web/API/NodeList>

querySelector

- Returnerar det första elementet som matchar.
- Samma syntax som CSS.
- Let el = document.querySelector(".myclass");
- querySelectorAll returnerar alla matchande element.

```
var highlightedItems =  
document.querySelectorAll(".highlighted");  
  
highlightedItems.forEach(function(userItem) {  
  deleteUser(userItem);  
}) ;
```

Selektorer

- Hur skulle ni kunna använda dessa funktioner för att effektivisera era beräkningsfunktioner?
- För den första summan skulle man t ex kunna ge alla rader en klass och hämta ut t ex ettorna för en spelare genom querySelectorAll().

```
<tr class="ones partsum1">  
  <td><input type="number" class="player1"></td>  
</tr>
```

```
document.querySelectorAll(".partsum1 .player1");
```

getAttribute / setAttribute

```
// getAttribute
var div1 = document.getElementById("div1");
var align = div1.getAttribute("align");

alert(align); // shows the value of align for the element
              // with id="div1"

// setAttribute
var b = document.querySelector("button");

b.setAttribute("name", "helloButton");
b.setAttribute("disabled", "");
```

parent / children

- If the node has no element children, then children is an empty list with a length of 0.

```
if (node.parentElement) {  
    node.parentElement.style.color = "red";  
}
```

Lambda-funktioner

- Kort repetition

```
// Jämför  
[1,2,3,4].filter(function (value) {  
    return value % 2 === 0  
}) ;
```

```
// med:  
[1,2,3,4].filter(value => value % 2 === 0);
```

Map



```
const myArray = [1,2,3,4];

const myArrayTimesTwo = myArray.map((value, index, array) => {
    return value * 2;
});

console.log(myArray); // [1,2,3,4];
console.log(myArrayTimesTwo); // [2,4,6,8];
```

- `map()`-metoden skapar en ny array med resultatet av att anropa en funktion för varje array-element.
- `map()`-metoden anropar den angivna funktionen en gång per element i arrayen i ordning.

Filter



```
const myArray = [1,2,3,4];

const myEvenArray = myArray.filter((value, index, array) => {
  return value % 2 === 0;
});

console.log(myArray); // [1,2,3,4];
console.log(myEvenArray); // [2, 4];
```

- `filter` tar emot samma argument som `map` och fungerar liknande. Enda skillnaden är att callback-funktionen måste returnera `true` eller `false`. Om den returnerar `true` kommer arrayen att behålla elementet, om den returnerar `false` kommer det att filtreras bort.

Reduce



```
const myArray = [1,2,3,4];

const sum = myArray.reduce((acc, currValue, currIndex, array) => {
    return acc + currValue;
}, 0);

const avg = sum / myArray.length;

console.log(avg); // 2.5
```

- reduce tar en array och reducerar den till ett enda värde.
Du kan t ex använda det för att räkna ut medelvärdet av alla värden i arrayen.

Map, Filter, Reduce

- Map: Hämta alla värden från siffrorna 1-6 till en HTMLCollection. Skapa en ny array med alla värden från arrayen.
https://jsfiddle.net/emmio_micke/Lwxm5814/11/
- Filter: Skapa en HTMLCollection från ett par kryssrutor. Skapa en ny array med de checkboxar som är ikryssade.
- Reduce: Hämta alla värden från siffrorna 1-6 till en HTMLCollection. Räkna ihop summan mha reduce.

YATZY	
SPELARE:	
Ettor	
Tvåor	
Treor	
Fyror	
Femmor	
Sexor	
SUMMA:	
BONUS (50)	
Par	
Två Par	
Triss	
Fyrtal	
Kåk	
Liten stege	
Stor stege	
Chans	
Vatzy (50)	
SUMMA:	

Högsta poäng
1 x 5 = 5p
2 x 5 = 10p
3 x 5 = 15p
4 x 5 = 20p
5 x 5 = 25p
6 x 5 = 30p
105poäng är max
Över 63p = 50 bonus
6+6 = 12p
6+6+5+5 = 22p
6+6+6 = 18p
6+6+6+6 = 24p
6+6+6+5+5 = 28p
1,2,3,4,5 = 15p
2,3,4,5,6 = 20P
6+6+6+6+6 = 30p
x+x+x+x+x = 50p
374p är max

Yatzy

- Gör om era funktioner som gör beräkningen för att uppdatera summan. Använd de array-funktioner ni har fått lära er.
- Gör ett tärningsformulär med fem text-rutor, fem kryssrutor och en knapp. När man klickar ska de rutorna som inte är kryssade få ett nytt tal.

```
let slump = Math.random();  
console.log(slump);
```

CSS Preprocessors

T ex SCSS & Less

- Är CSS ett programmeringsspråk?
- Preprocessorer används för att utöka möjligheterna med CSS, t ex:
 - Variabler
 - Nesting
 - Mixins
 - Loopar
 - Arv
- Olika för olika preprocessorer
- Kompileras till css

SCSS Variabler

```
$primary-color: #3bbfce;  
$margin: 16px;  
  
.content-navigation {  
    border-color: $primary-color;  
    color: darken($primary-color, 10%);  
}  
  
.border {  
    padding: $margin / 2;  
    margin: $margin / 2;  
    border-color: $primary-color;  
}  
  
.content-navigation {  
    border-color: #3bbfce;  
    color: #2b9eab;  
}  
  
.border {  
    padding: 8px;  
    margin: 8px;  
    border-color: #3bbfce;  
}
```

SCSS

Nesting

```
table.hl {  
  margin: 2em 0;  
  td.ln {  
    text-align: right;  
  }  
}  
  
li {  
  font: {  
    family: serif;  
    weight: bold;  
    size: 1.3em;  
  }  
}
```

```
table.hl {  
  margin: 2em 0;  
}  
table.hl td.ln {  
  text-align: right;  
}  
  
li {  
  font-family: serif;  
  font-weight: bold;  
  font-size: 1.3em;  
}
```

SCSS Mixins

```
@mixin table-base {  
  th {  
    text-align: center;  
    font-weight: bold;  
  }  
  td, th {  
    padding: 2px;  
  }  
}  
  
#data {  
  @include table-base;  
}
```

```
#data th {  
  text-align: center;  
  font-weight: bold;  
}  
#data td, #data th {  
  padding: 2px;  
}
```

SCSS Loopar

```
$squareCount: 3
@for $i from 1 through $squareCount
  #square-#{$i}
    background-color: red
    width: 50px * $i
    height: 120px / $i
```

```
#square-1 {
  background-color: red;
  width: 50px;
  height: 120px;
}

#square-2 {
  background-color: red;
  width: 100px;
  height: 60px;
}

#square-3 {
  background-color: red;
  width: 150px;
  height: 40px;
}
```

SCSS

Arv

```
.error
  border: 1px #f00
  background: #fdd

.error.intrusion
  font-size: 1.3em
  font-weight: bold

.badError
  @extend .error
  border-width: 3px
```

```
.error, .badError {
  border: 1px #f00;
  background: #fdd;
}

.error.intrusion,
.badError.intrusion {
  font-size: 1.3em;
  font-weight: bold;
}

.badError {
  border-width: 3px;
}
```

SCSS

- *Uppgift:* Installera stöd för SCSS i Visual Studio Code.
- https://code.visualstudio.com/docs/languages/css#_transpiling-sass-and-less-into-css