

# 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.

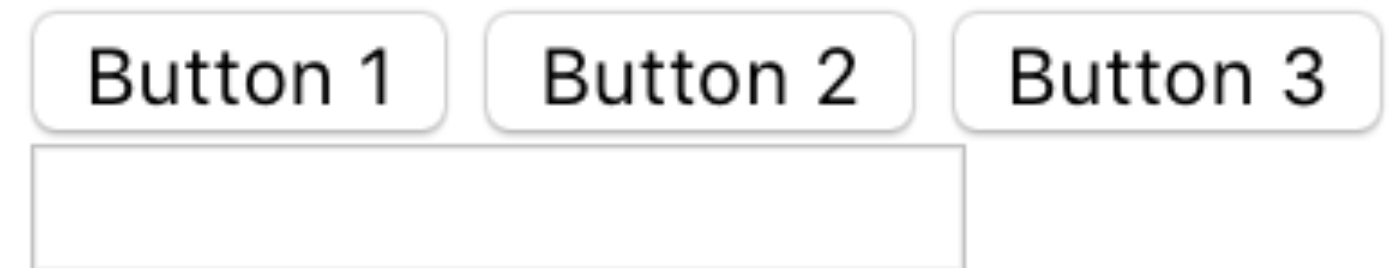
```
let slump = Math.random();  
console.log(slump);
```



Button 1 Button 2 Button 3

# 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!");  
  })  
});
```

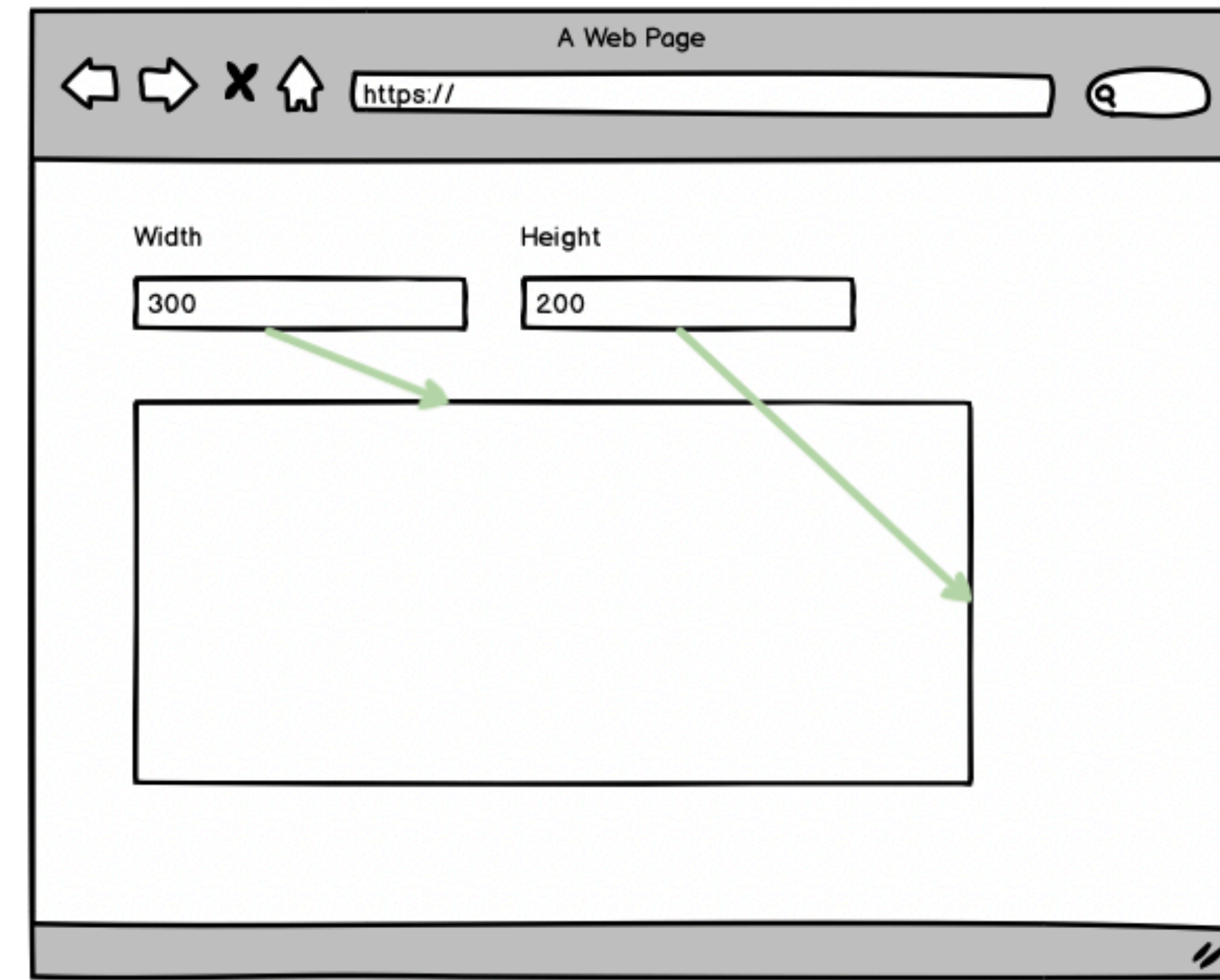


# Andra event-typer

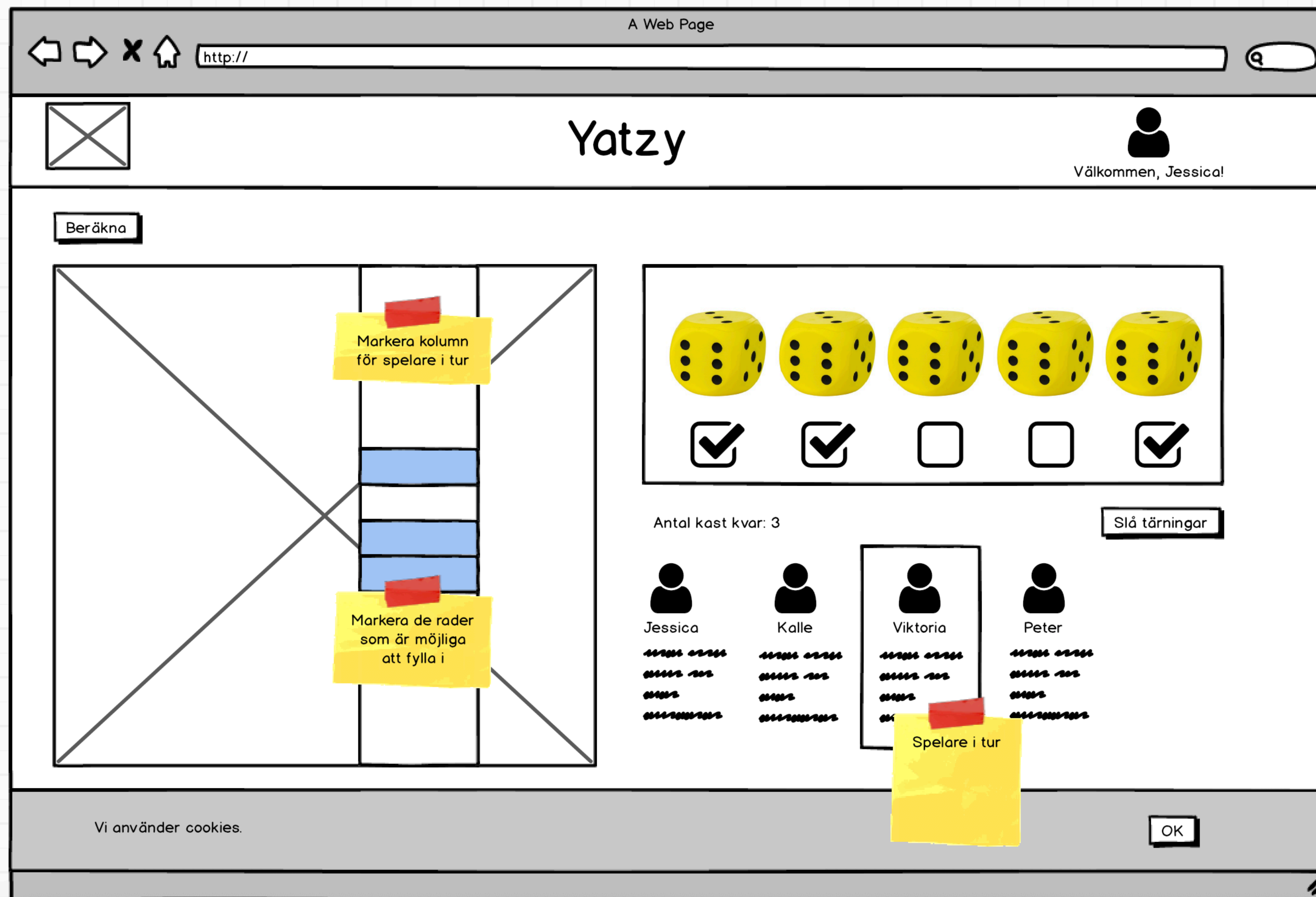
- `keypress`
- `keydown`
- `keyup`
- `click`
- `mousedown`
- `mouseup`
- `change`
- `submit`
- <https://developer.mozilla.org/en-US/docs/Web/Events>

# 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.



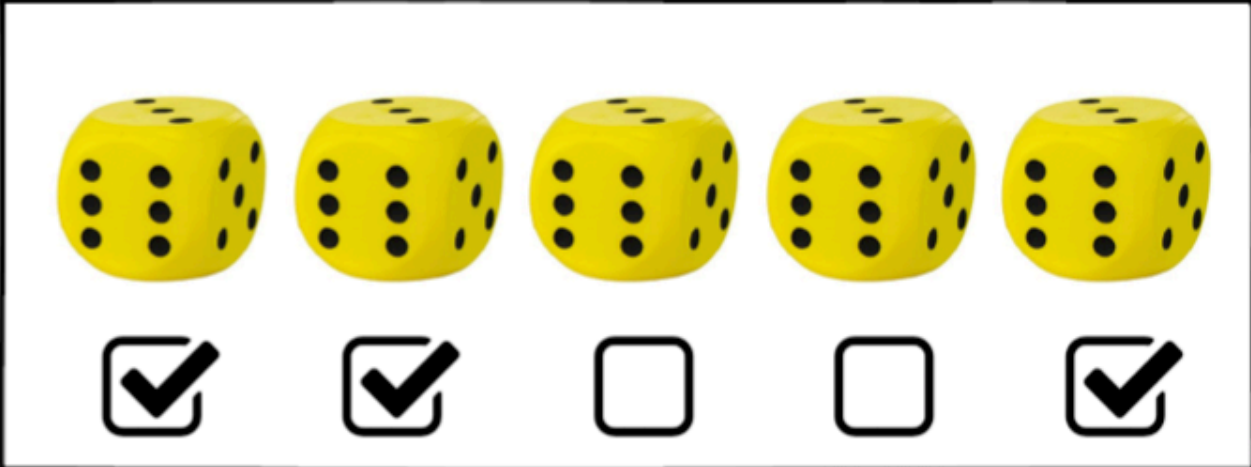
# Yatzy - layout



# 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);
```



Antal kast kvar: 3

Slå tärningar

# 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);  
}
```



# getElementsByTagName

- 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");
```

- [https://jsfiddle.net/emmio\\_micke/aopLq2gt/](https://jsfiddle.net/emmio_micke/aopLq2gt/)

# 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", "");
```

# innerHTML / textContent

- textContent har bättre prestanda eftersom dess innehåll inte parses som HTML. Dessutom kan textContent förhindra XSS attacker.
- Cross-site scripting (XSS) är ett säkerhetshål som låter någon injicera skadlig kod. Denna kod körs av klienten.
- <https://www.santor.se/kunskapsbank-it-sakerhet/appsakerhet/vad-innebar-cross-site-scripting-xss/>

```
// innerHTML
var div1 = document.getElementById("div1");
div1.innerHTML = "New content";
```

# parent / children

- If the node has no element children, then children is an empty list with a length of 0.
- <https://developer.mozilla.org/en-US/docs/Web/API/Node>

```
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/](https://jsfiddle.net/emmio_micke/Lwxm5814/)

- 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:						Högsta poäng
Ettor						1 x 5 = 5p
Tvåor						2 x 5 = 10p
Treor						3 x 5 = 15p
Fyror						4 x 5 = 20p
Femmor						5 x 5 = 25p
Sexor						6 x 5 = 30p
SUMMA:						105poäng är max
BONUS (50)						Över 63p =50 bonus
Par						6+6 = 12p
Två Par						6+6+5+5 = 22p
Triss						6+6+6 = 18p
Fyroral						6+6+6+6 = 24p
Kör						6+6+6+5+5 = 28p
Liten stege						1,2,3,4,5 = 15p
Stor stege						2,3,4,5,6 = 20P
Chans						6+6+6+6+6 = 30p
Yatzy (50)						x+x+x+x+x = 50p
SUMMA:						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](https://code.visualstudio.com/docs/languages/css#_transpiling-sass-and-less-into-css)

# Styra utseende med JS

- Man kan styra enskilda style properties.
  - `document.getElementById("something").style.backgroundColor = "#ccdd33";`
- [https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\\_Properties\\_Reference](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Properties_Reference)

# CSS vs JS

- Vi kan t ex sätta en styling på ett aktuellt element när det kommer i fokus.
- Det kan vi göra med CSS också.
- Med JS är det enklare att styra andra elements utseende.
- [https://jsfiddle.net/emmio\\_micke/r8uz1bdm/](https://jsfiddle.net/emmio_micke/r8uz1bdm/)

# Styra utseende med JS

- Man kan även lägga till eller ta bort klasser.

- `document.getElementById("div1").classList.add("classToBeAdded");`
- `document.getElementById("div1").classList.remove("classToBeRemoved");`

- Man kan togglar klasser.

- `document.getElementById("div1").classList.toggle("classToBeToggled");`

- Och kontrollera om en klass finns.

- ```
if (document.getElementById("div1").classList.contains("classToBeChecked")) {  
    // Do something  
}
```

- [https://www.kirupa.com/html5/setting\\_css\\_styles\\_using\\_javascript.htm](https://www.kirupa.com/html5/setting_css_styles_using_javascript.htm)

# Styra utseende med JS

- I era Yatzy-spel:
  - Skapa en klass som gör det tydligt att en "poäng" (t ex raden för fyrtal eller kåk) är otillgänglig.
  - Visa vilken spelare som är aktiv.
  - [https://jsfiddle.net/emmio\\_micke/njtdfrkz/](https://jsfiddle.net/emmio_micke/njtdfrkz/)