**Realisieren**

Es ist wichtig als erstes eine Funktion zu haben, welche alle möglichen Felder einer Figur in einem Array abspeichert:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung  
Die Funktion schaut was für eine Art Figur es ist, wonach die möglichen Felder berechnet und zurückgegeben werden.

Jetzt mussten wir wählen, wie wir unsere Funktionen zum Bewegen der Figur abgerufen werden.  
Entschieden haben wir uns für ein JSON-Objekt, welches nötige Variablen und die dazugehörige Funktion beinhaltet:  
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung  
InteractWithPiece() erfüllt hierbei das Befüllen der möglichen Felder mit Punkten. Für jede Figur sieht die Funktion etwas anders aus.  
Diese Funktion wäre zum Beispiel für die Felder des Läufers, welche in Richtung oben links gehen:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Die Punkte, welche durch diese Funktionen erstellt werden, brauchen jetzt einen EventListener, das machten wir mit folgender Funktion:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

In einigen if-Statements sieht man die Variable moveTest, diese ist eine Vorbereitung auf eine Zukünftige Überprüfung der angegriffenen Felder.

Bis hierhin brauchten wir nur Quellen, die uns halfen beim CSS eines Kreises. Denn wir wollten, dass die Kreise sich im Vordergrund von den Figuren befinden.

Hier sind die Quellen:  
<https://www.w3schools.com/howto/howto_css_circles.asp>  
<https://www.tutorialrepublic.com/faq/how-to-overlay-one-div-over-another-div-using-css.php>

Unsere Zeit reichte nicht vollständig aus um sich um die Sonderregeln zu kümmern.

Wir planet das nächste Mal in zwei Gruppen zu arbeiten, eine welche sich um das Schach, das Schachmatt und das Remis kümmert und eine, welche die Sonderregeln übernimmt.

Wir bemerkten, dass das zeitlich recht knapp wird, also entschlossen wir uns am Wochenende die Sonderregeln zu implementieren.

Jetzt galt es En Passant, Rochade und die Bauernumwandlung zu implementieren.

Dieser Code legt fest ob ein Bauer von en Passant betroffen ist:Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Hier wird dann für einen gegnerischen Bauern festgelegt ob er jemanden durch en Passant angreifen kann:  
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Dieser Code legt fest ob gerade ein Bauer umgewandelt werden kann und zeigt, dann die möglichen Umwandlungen an:Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Für die Rochade wird überprüft, ob sich Turm und König schon mal bewegt haben und ob die Felder zwischen den beiden angegriffen werden, im Moment haben wir aber keine Funktion, welche überprüft, welche Felder unter Angriff stehen, das muss nächstes Mal erledigt werden. Die Werte hier sind lediglich eine Vorbereitung:  
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Bei jedem klick auf einen Bewegungspunkt wird überprüft, ob der Punkt die id „castle“ oder „longcastle“ hat. So weiss das Skript dass es den König und den Turm bewegen muss:Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Der erste Schritt, um überprüfen zu können, ob im Moment der König im Schach ist, ist eine Funktion welche alle vom Gegner angegriffenen Felder zurückgibt.  
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

typeOfTest befüllt nun die Variable moveTest, welche wir im Voraus vorbereitet haben.

Nach jedem MoveTest werden die Felder mit Punkten in ihnen aktualisiert und als attackiert markiert:  
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung