



# Agenda

- ▶ The style tag and CSS (Cascading style sheet)
- ▶ CSS Selectors
- ▶ style tag
- ▶ class attribute
- ▶ position:
  - static
  - fixed
  - absolute
  - relative.

# HTML is not a programming language

## Before we continue

- ▶ Like a java program, A web site is code in the form of text...
- ▶ But, unlike a java program, This text is usually located on a remote computer and instead of a compiler, the program that is in charge of interpreting it is the browser
- ▶ HTML is more like data representation than a programming language



# Styling

- ▶ The style attribute is a very important attribute in HTML



- ▶ There are many different style properties. In this lecture, we will go through some basic concepts in website styling
- ▶ For more style attributes, see [here](#).

# New Job

▶ Congratulations! you got your first job at a company called "Playtrix"



## Job title:

Mockups implementer

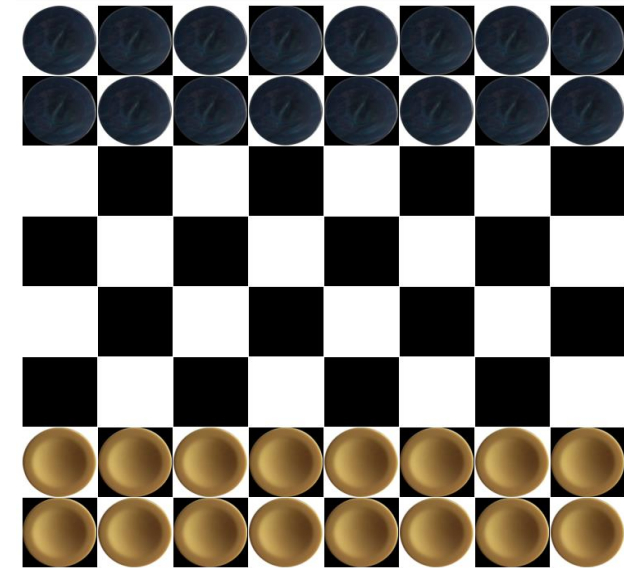
## Job description:

The designer will send you an image of a required page (created with Photoshop) and you will convert it into HTML with an advanced set of tools (your head and two hands)



# New Job

- ▶ Playtrix is a gaming company focused on casual games



- ▶ Your first job will be to implement the basic HTML page for one of their free to play games:
- ▶ Our weird version of Checkers

- ▶ Going back to our checkers board from the guided exercise, You remember that it was a tedious job of copy and paste
- ▶ But, you are not afraid of hard work... so:

# Styling

- ▶ After a long two hours of work (Typing code, copy & paste and validating the results of course...)
- ▶ The Designer comes to your desk...

We just had a meeting about Checkers... we decided to make the squares smaller, so please change it to 80px by 80px



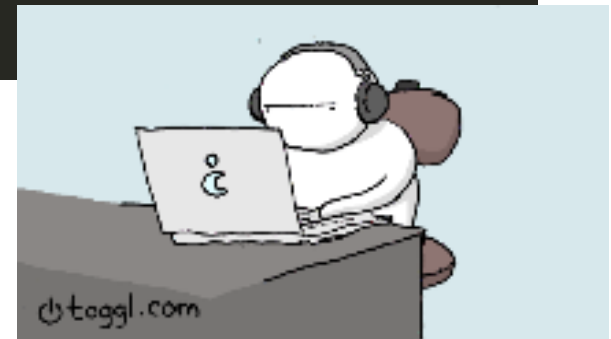
Can anyone name him please?



# So what do you do?

[illegible]

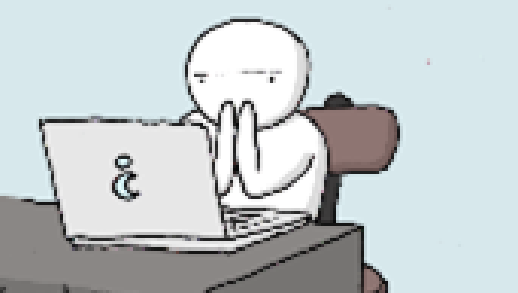
# Search & replace...



# Styling

Hey new guy, 80px is too small..  
Make it 90px, Oh and please add 6px,  
black border to the board itself





# Styling

The CMO just finished her research and it turns out that if we replace the white squares with red, the players stay longer. And also the front end dev requested a different mouse cursor for the pieces



```
<img style="width:90px;height:90px;cursor:pointer;"
```

Urrggghhh!



# Styling

- ▶ You probably think to yourself "Surely there must be an easier way..."

There is!  
It is called CSS (cascading style sheets)



# Styling

Meet the `<style>` tag

`<style></style>`

The `<style>` tag is used to define style **information** for an HTML document.

The style tag must be located inside the `<head>` tag of your document



# Styling

▶ Meet the `<style>` tag

`<style></style>`



Inside the `<style>` element you specify **how** HTML elements should render in a browser.



Each HTML document can contain multiple `<style>` tags.





# Styling

▶ For example:

```
<style>
  img{
    width:90px;
    height:90px;
    cursor:pointer;
  }
</style>
```

▶ Now, all of our <img> tags will receive those styling properties

```
tagname {
  propname:propvalue;
  propname2:propvalue2;
}
```

Notice the syntax!

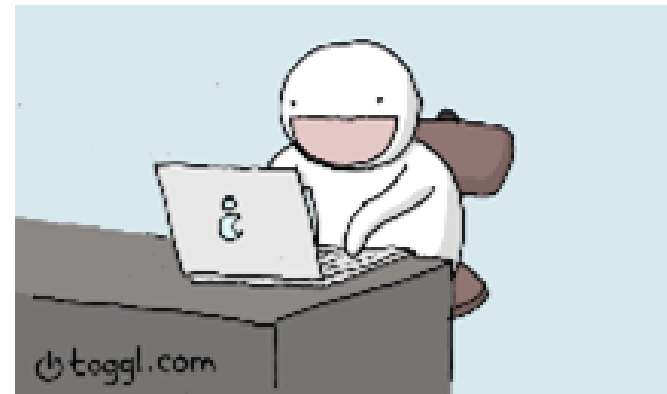


# Styling

- ▶ Now we can remove all of the style attributes from our `<img>` tags in the document

```
<td style="background-color:red;width:90px;height:90px;"></td>
<td style="background-color:black;width:90px;height:90px;"></td>
<td style="background-color:red;width:90px;height:90px;"></td>
```

- ▶ Much better!



# Styling

- ▶ And also move the common style properties from the `<td>` tag to the `<style>` tag

```
<style>
  img{
    width:90px;
    height:90px;
    cursor:pointer;
  }
  td{
    width:90px;
    height:90px;
  }
</style>
```

- ▶ The code is much cleaner now

```
<table style="border-width:6px;border-color:black;border-style: solid;">
  <tr>
    <td style="background-color:red;"></td>
    <td style="background-color:black;"></td>
    <td style="background-color:red;"></td>
    <td style="background-color:black;"></td>
    <td style="background-color:red;"></td>
    <td style="background-color:black;"></td>
    <td style="background-color:red;"></td>
    <td style="background-color:black;"></td>
  </tr>
  <tr>
    <td style="background-color:black;"></td>
    <td style="background-color:red;"></td>
    <td style="background-color:black;"></td>
    <td style="background-color:red;"></td>
    <td style="background-color:black;"></td>
    <td style="background-color:red;"></td>
    <td style="background-color:black;"></td>
    <td style="background-color:red;"></td>
  </tr>
  <tr>
    <td style="background-color:red;"></td>
    <td style="background-color:black;"></td>
    <td style="background-color:red;"></td>
    <td style="background-color:black;"></td>
```



# Styling

Let's change the square size back to 100px

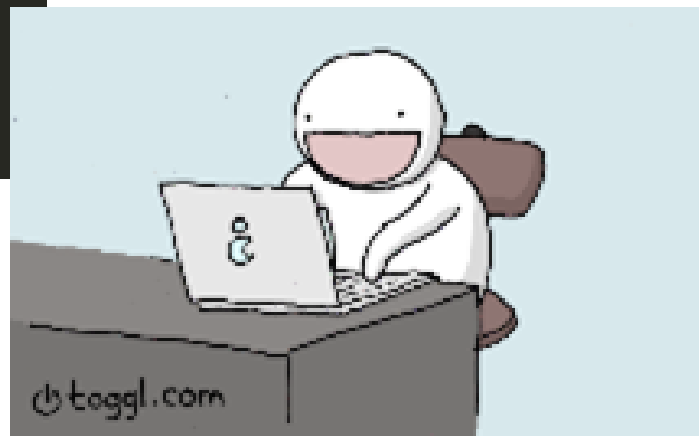


# Styling

But now can you smile!

No problem!

```
<style>
  img{
    width:100px;
    height:100px;
    cursor:pointer;
  }
  td{
    width:100px;
    height:100px;
  }
</style>
```



# Questions?



# Styling

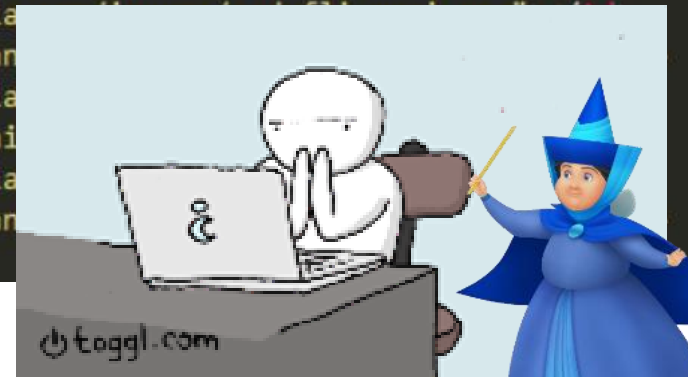
Hey, Melissa does not like the default red color you chose.  
Please use this one: `rgb(200,20,60)` it is called Crimson



# Styling

- ▶ We need a way to group all of the red cells and tell the browser to set their background to the new red (I mean Crimson)

```
<tr>
  <td style="background-color:red;"></td>
  <td style="background-color:black;"></td>
  <td style="background-color:red;"></td>
  <td style="background-color:black;"></td>
  <td style="background-color:red;"></td>
  <td style="background-color:black;"></td>
  <td style="background-color:red;"></td>
  <td style="background-color:black;"></td>
</tr>
```



CSS can help us with that too!

**Class attribute:** We can use a new attribute,  
The class attribute is used to group HTML elements that have a common purpose or meaning.



# Styling

- ▶ We add the class attribute like any other HTML attribute
- ▶ Try to give it a meaningful name
- ▶ We will use "dark" for the black and "light" for the "red". This is how it looks:

```
<td class="dark"></td>
<td class="light"></td>
<td class="dark"></td>
<td class="light"></td>
<td class="dark"></td>
<td class="light"></td>
<td class="dark"></td>
<td class="light"></td>
```

- ▶ And then add the class definition to the <style> tag

```
.dark {
    background-color: black;
}

.light {
    background-color: rgb(200, 20, 60)
}
```

# Styling

Our new code:

```
<style>
  img {
    width: 100px;
    height: 100px;
    cursor: pointer;
  }

  td {
    width: 100px;
    height: 100px;
  }

  .dark {
    background-color: black;
  }

  .light {
    background-color: rgb(200,20,60)
  }
</style>
```

▶ All <img> tags in the document should be 100x100 and have a pointer cursor

▶ All <td> tags in the document should be 100x100

▶ All elements in the document that are classified as dark should be black

▶ All elements in the document that are classified as "light" should be crimson

# Summary

▶ CSS separates presentation(style) from content(data)



- In order to connect between the two we use the class attribute

# Questions?



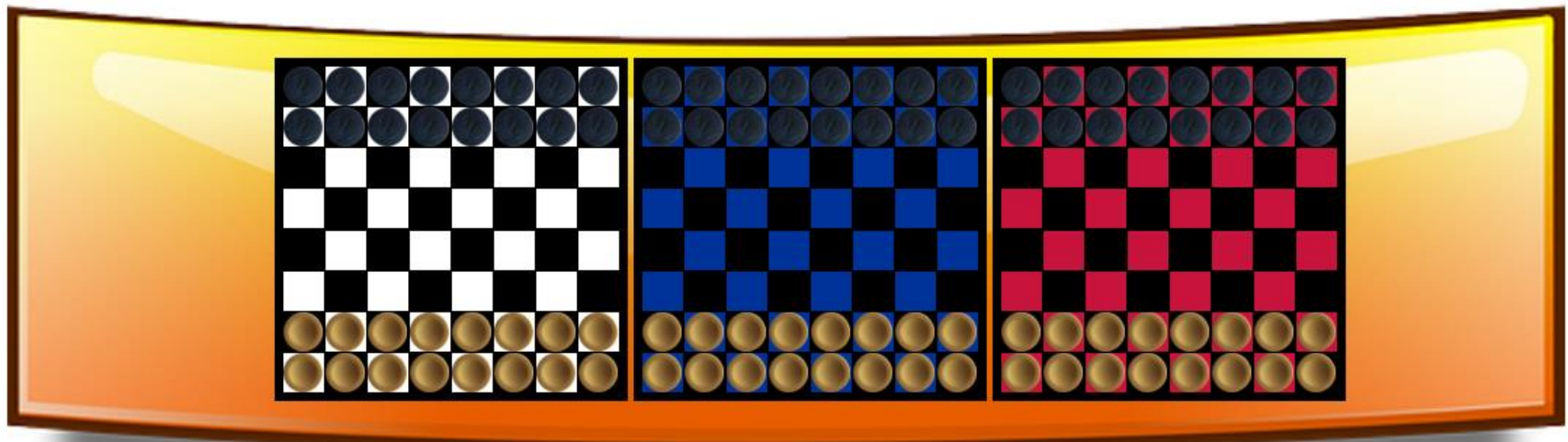
# Styling

Hey,  
We are building the website for the  
checkers game.  
Here is the mockup...

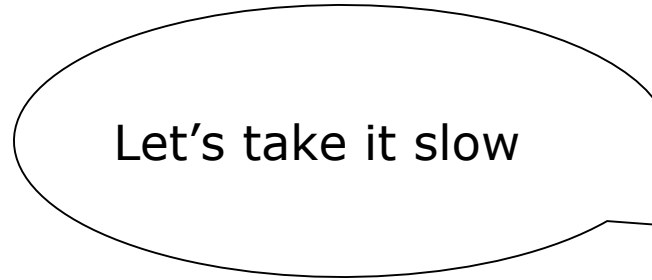


# Styling

## Welcome To Checkers

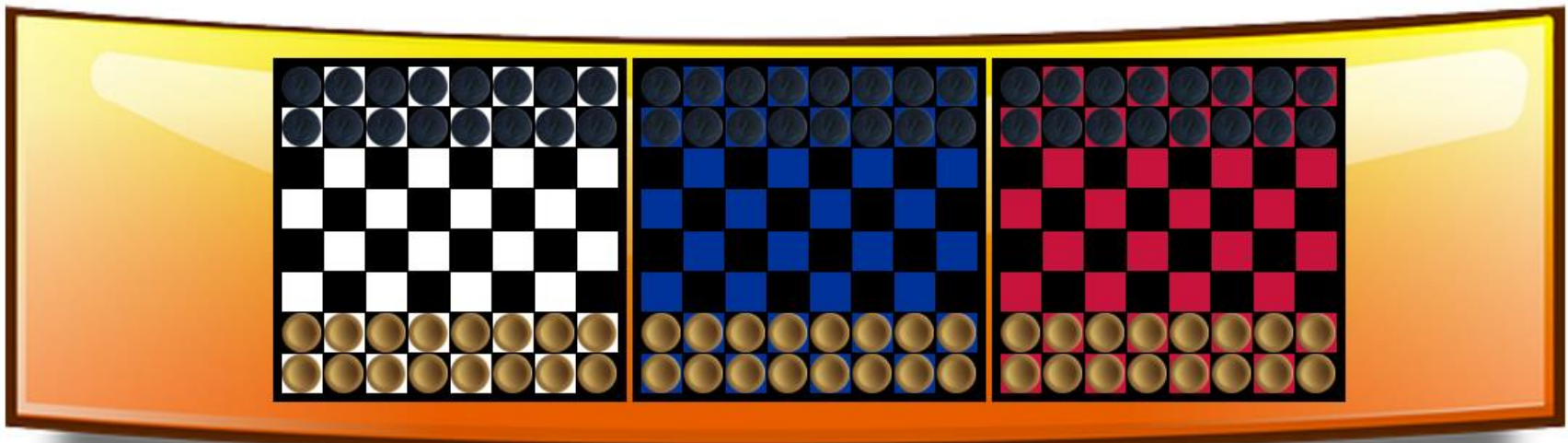


# Styling



# Styling

## Welcome To Checkers



- ▶ How can we create the three different boards?
- ▶ First, we need to copy and paste the board's HTML structure.





# Styling

▶ We will add a new class attribute to each table



```
<table class="white-table">
  <tr>
    <td class="dark"></td>
    <td class="light"></td>
```

```
<table class="blue-table">
  <tr>
    <td class="dark"></td>
    <td class="light"></td>
```

```
<table class="red-table">
  <td class="dark"></td>
  <td class="light"></td>
```

# Styling

▶ And add the following css rules to our <style> tag



```
.white-table .light{
  background-color: white;
}

.blue-table .light{
  background-color: blue;
}

.red-table .light{
  background-color: red;
}
```

All elements classified as "light" under an element classified as "white-table" Should have a white background

# Styling

## CSS Selectors



In CSS, selectors are patterns used to select the element(s) you want to style.

```
.white-table .light{
  background-color: white;
}

.blue-table .light{
  background-color: blue;
}

.red-table .light{
  background-color: red;
}
```

.white-table .light  
is a **css selector**

# Styling

## CSS Selectors



#id	<i>example: #itc-12345678</i>
.class	<i>example: .white-table</i>
[attribute=value]	<i>example: [src='www.israeltechallenge.com/icon1.png']</i>
HTML tag	<i>example: div</i>

*More?*

[http://www.w3schools.com/cssref/css\\_selectors.asp](http://www.w3schools.com/cssref/css_selectors.asp)

# Styling

- ▶ This is a new CSS selector (which is actually a single space)
- ▶ It describes the relation of a parent element and a child element

space



```
parent-tagname.parent-classname child-tagname.classname {
    propname:propvalue;
}
```

```
<parent-tagname class="parent-classname">
  <tagname class="classname"></tagname>
</parent-tagname>
```

# Styling

## CSS Selectors – Relations



<u>element</u>	p	Selects all <p> elements
<u>element,element</u>	div, p	Selects all <div> elements and all <p> elements
<u>element element</u>	div p	Selects all <p> elements inside <div> elements
<u>element&gt;element</u>	div > p	Selects all <p> elements where the parent is a <div> element
<u>element+element</u>	div + p	Selects all <p> elements that are placed immediately after <div> elements
<u>element1~element2</u>	p ~ ul	Selects every <ul> element that are preceded by a <p> element

# Styling

▶ The child can be any descendant of the parent tag!

```
<div class="container">
  <div class="wrapper">
    <table>
      <tr><td><h3>hello1</h3><td></tr>
      <tr><td><h3>hello2</h3><td></tr>
      <tr><td><h3>hello3</h3><td></tr>
    </table>
  </div>
</div>
```

```
.container h3{
  color:red;
}
```



# Styling

- ▶ The child can be any descendant of the parent tag!

```
<div class="container">
  <div class="wrapper">
    <table>
      <tr><td><h3>hello1</h3><td></tr>
      <tr><td><h3>hello2</h3><td></tr>
      <tr><td><h3>hello3</h3><td></tr>
    </table>
  </div>
</div>
```



```
.container h3{
  color:red;
}
```



# Styling

The class attribute specifies one or more class names for an element.

```
.classname {  
    propname:propvalue;  
}
```

# Styling

- ▶ The class attribute specifies **one or more** class names for an element.

What does it mean?

It means that an HTML element can have 2 classnames.

Separated by a space

HTML:

```
<img class="classname1 classname2" />
```

no space

CSS:

```
.classname1.classname2 {
    propname:propvalue;
}
```

# Styling

## CSS Selectors – Game Time

```
<body>
  <div class="menu">
    PLAYGROUND
    <span class="menu-item">about</span>
    <span class="menu-item">contact</span>
    <span class="menu-item">home
      
    </span>
    <a href="http://www.google.com">google</a>
  </div>
  <div class="pages">
    <span>
      
      <a href="http://www.w3.org">w3c</a>
    </span>
  </div>
</body>
```

How to style  
this element?

Css selector	example
Tag selector	div
Class selector	.my-class
Attribute	[target="self"]
Nesting	div span

# Styling

## CSS Selectors – Game Time

```
<body>
  <div class="menu">
    PLAYGROUND
    <span class="menu-item">about</span>
    <span class="menu-item">contact</span>
    <span class="menu-item">home
      
    </span>
    <a href="http://www.google.com">google</a>
  </div>
  <div class="pages">
    <span>
      
      <a href="http://www.w3.org">w3c</a>
    </span>
  </div>
</body>
```

How to style  
this element?

Css selector	example
Tag selector	div
Class selector	.my-class
Attribute	[target="self"]
Nesting	div span

# Styling

## CSS Selectors – Game Time

```
<body>
  <div class="menu">
    PLAYGROUND
    <span class="menu-item1">about</span>
    <span class="menu-item2">contact</span>
    <span class="menu-item3">home
      
    </span>
    <a href="http://www.google.com">google</a>
  </div>
  <div class="pages">
    <span>
      
      <a href="http://www.w3.org">w3c</a>
    </span>
  </div>
</body>
```

How to style  
this element?



Css selector	example
Tag selector	div
Class selector	.my-class
Attribute	[target="self"]
Nesting	div span

# Styling

## CSS Selectors – Game Time

```
<body>
  <div>
    PLAYGROUND
    <span class="menu-item bold">about</span>
    <span class="menu-item">contact</span>
    <span class="menu-item">home
      
    </span>
    <a href="http://www.google.com">google</a>
  </div>
  <div class="pages">
    <span>
      
      <a href="http://www.w3.org">w3c</a>
    </span>
  </div>
</body>
```

How to style  
this element?

Css selector	example
Tag selector	div
Class selector	.my-class
Attribute	[target="self"]
Nesting	div span

# Styling

## CSS Selectors – Game Time

```
<body>
  <div>
    PLAYGROUND
    <span class="menu-item">about</span>
    <span class="menu-item">contact</span>
    <div class="menu-item">home
      
    </div>
    <a href="http://www.google.com">google</a>
  </div>
  <div>
    <span>
      
      <a href="http://www.w3.org">w3c</a>
    </span>
  </div>
</body>
```

How to style  
this element?

Css selector	example
Tag selector	div
Class selector	.my-class
Attribute	[target="self"]
Nesting	div span

# Styling

Remember me?

Welcome To Checkers



- ▶ Let's get back to our game
- ▶ We dealt with colors, but also the rows should be smaller...



# Styling

▶ We will also reduce the cell size to 30px

```
img{  
  width:30px;  
  height:30px;  
  cursor:pointer;  
}  
  
td{  
  width:30px;  
  height:30px;  
}
```

# So far...

- ▶ We copied our tables
- ▶ Added colors:

```
.white-table .light{
  background-color: white;
}

.blue-table .light{
  background-color: blue;
}

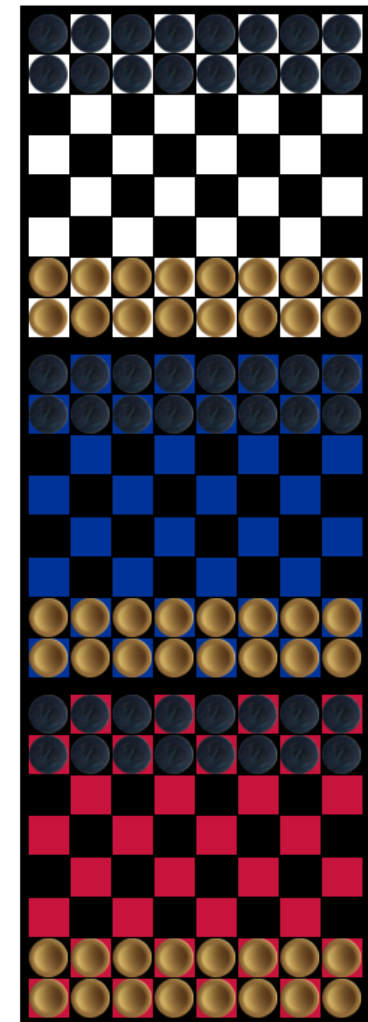
.red-table .light{
  background-color: red;
}
```

- ▶ And changed the size:

```
img{
  width:30px;
  height:30px;
  cursor:pointer;
}

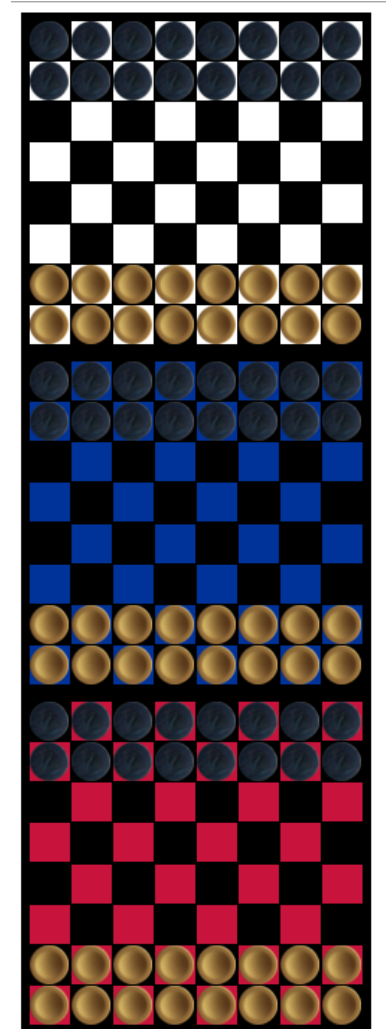
td{
  width:30px;
  height:30px;
}
```

The result:



# Styling

- ▶ The boards are sitting on top of each other But in the mockup they are aligned in a row
- ▶ The reason: The "display" property
- ▶ A very important property that defines the layout of elements in the doc
- ▶ It has 4 basic values:
  - Inline
  - Inline-block
  - Block
  - none



# Styling

▶ As mentioned in the last lecture, every element has a default display value. However, you can override this with style.

▶ Some inline elements (do you remember?)

`<span>`

`<a>`

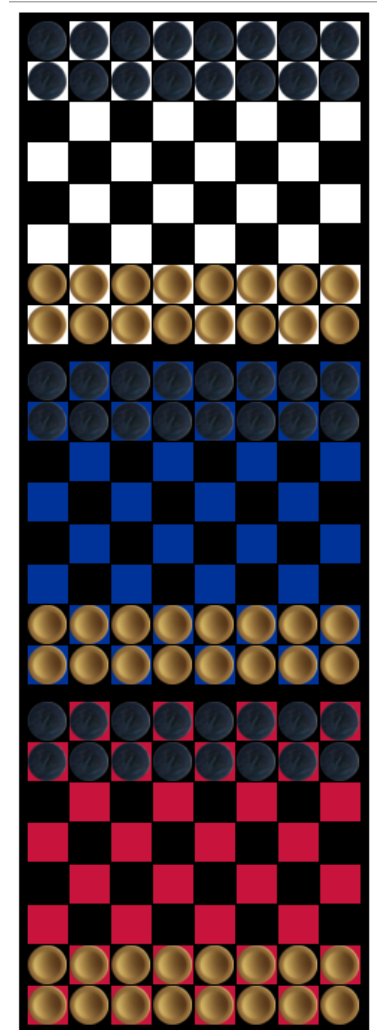
`<img>`

▶ Some block elements

`<div>`

`<h1>` - `<h6>`

`<p>`



# Inline Vs Block

## ▶ Content model

- Inline elements should only contain other inline elements as well as content
- block elements can contain block elements, inline elements and content

## ▶ Appearance

- Inline elements can't have width or height properties
- When they have no space limitation, Block elements can take up the entire screen

# Inline Vs Block



Inline elements can't have width or height properties?

What does it mean?

We encourage you not to take things for granted, but to test it yourselves!

# Inline Vs Block

## ▶ Formatting

- Inline elements do not begin on a new line (they can start everywhere)
- Block elements begin on a new line and take all of that line

```
<h1>Hello </h1>
```

```
<h2>How are you?</h2>
```

```
<span>Hope this webpage finds you well </span>
```

```
<a href="candycrushsaga.com/"> and ready for fun!</a>
```

**Hello**

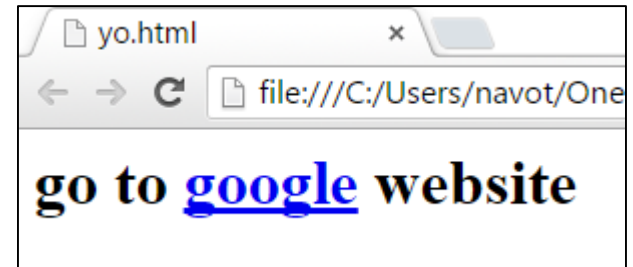
**How are you?**

Hope this webpage finds you well and ready for fun!

# Inline Vs Block

- ▶ Inline elements don't "disrupt" the layout of other elements next to it

```
<h1>go to <a href="www.google.com" >google</a> website</h1>
```



- ▶ Where as block elements...

```
<a href="www.google.com" > <h1>go to google</h1> </a> website
```





# Inline-block and None

- ▶ Display: none; means don't show the element and the content in it
- ▶ Display: inline-block; are like inline elements, but you can set their width and height.

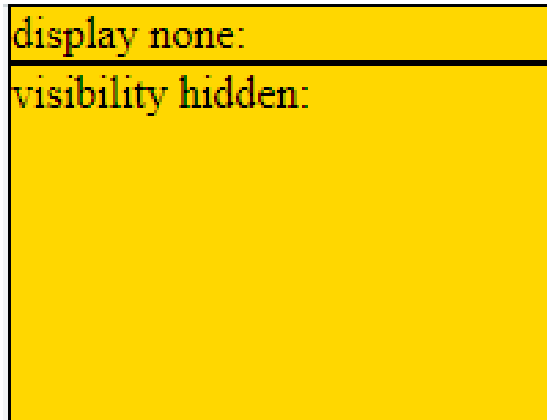
# Inline-block and None

- ▶ Display: none; means don't show the element and the content in it
- ▶ Not to be confused with visibility: hidden

## HTML:

```
<div>display none:
  <div class="inner display-none"></div>
</div>
<div>visibility hidden:
  <div class="inner visibility-hidden"></div>
</div>
```

[Link to code](#)



## CSS:

```
div {
  background: gold;
  border: solid black 1px;
  width: 200px;
}

.inner {
  height: 100px;
  background: salmon;
}

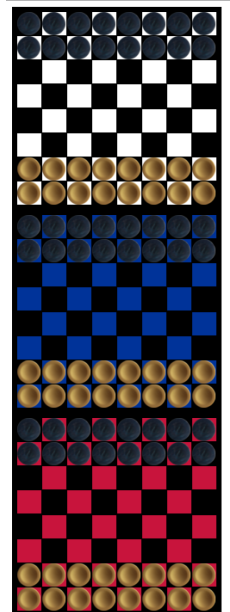
.display-none {
  display: none;
}

.visibility-hidden {
  visibility: hidden;
}
```

# Styling

Back to our example...

- ▶ A `<table>` tag can and should contain other block elements. This is why it is a block-level element
- ▶ Block-level elements end with a new line (a bit like `println`)
- ▶ Because this is just a formatting decision we can change it using style



# Styling

▶ Changing the formatting property value of an element will not change the content meaning of the block!!!

▶ An inline element with `display: block;` is not allowed to have other block elements inside it.



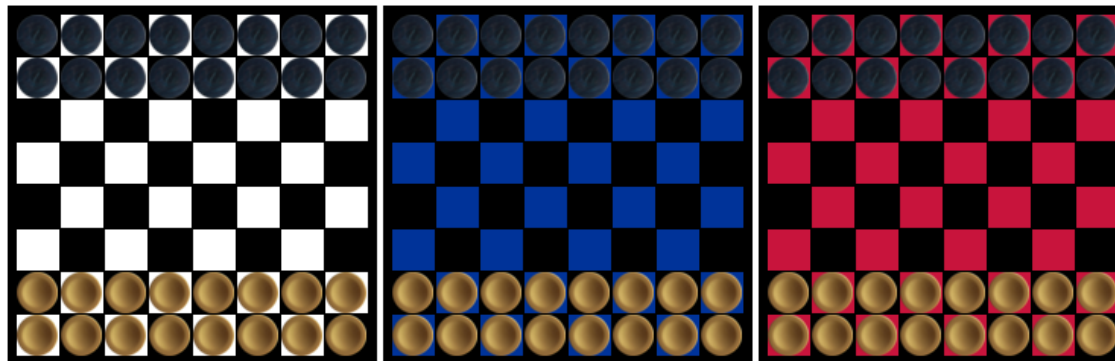
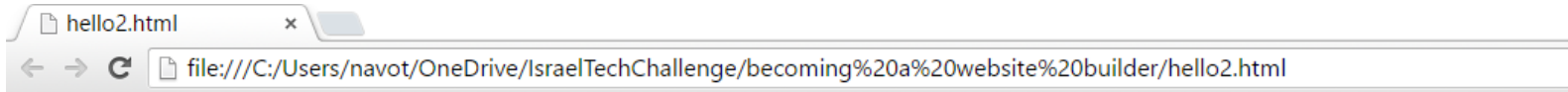
▶ It will still work though... but makes no sense

```
>dy>
  <a href="google.com" style="display:block">
    <div>sdsadad</div>
    <div>sdsadad</div>
```

sdsadad  
sdsadad

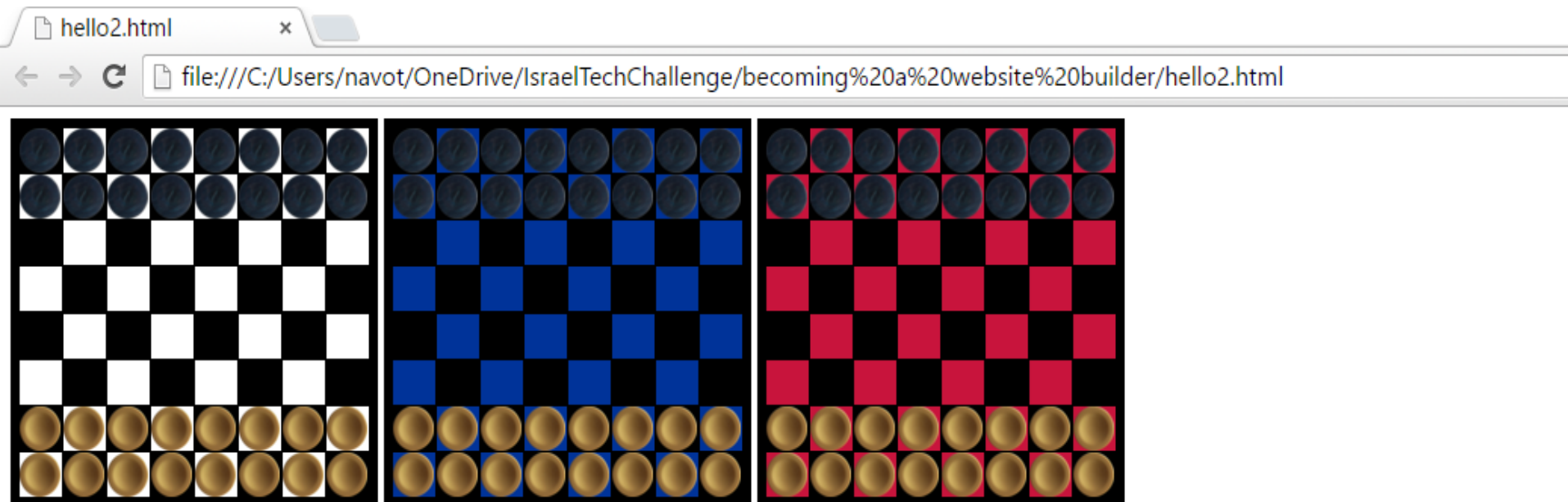
# Styling

```
table{
  border-width:6px;
  border-color:black;
  border-style: solid;
  background-color:black;
  display:inline;
}
```



# Styling

- ▶ We need to center the boards
- ▶ To do that we can use the text-align property
- ▶ The text-align property sets the align of children elements in a container

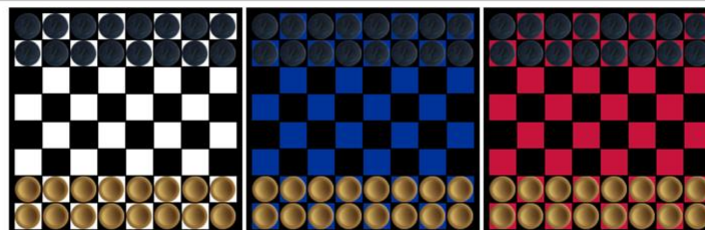


# Styling

- ▶ So we will wrap the boards with a `<div>` and name it "boards-holder"

```
<div class="boards-holder">
  <table class="white-table">
    <tr>
      <td class="dark"><img src="http://www.gammonmania.com/images/sede
```

```
.boards-holder{
  text-align:center;
}
```



# Styling

Adding the title is a piece of cake

## Welcome To Checkers





# Styling

```
<h1 class="game-title">Welcome To Checkers</h1>
<div class="boards-holder">
  <table class="white-table">
    <tr>
      <td class="dark">
<div class="image-holder">
  
  
</div>
```

What will be the width of this img?

```
img {
  width: 200px;
  height: 200px;
  border: 1px solid black;
}
```

Courtesy chrome dev tools



# CSS precedence

```
<style>
  img{
    width:200px;
    height:200px;
    border:1px solid black;
  }
  img.banner{
    width:500px;
    cursor:pointer;
  }
  .image-holder img.banner{
    width:150px;
  }
</style>
```

```

<div class="image-holder">
  
</div>
```

What will be the width of this img?

```
.image-holder img.banner { test.html:12
  width: 150px;
}
img.banner { test.html:8
  width: 500px;
  cursor: pointer;
}
img { test.html:3
  width: 200px;
  height: 200px;
  border: 1px solid black;
}
```

# CSS precedence

```
<style>
  img{
    width:200px;
    height:200px;
    border:1px solid black;
  }
  img.banner{
    width:500px;
    cursor:pointer;
  }
  .image-holder img.banner{
    width:150px;
  }
</style>
```

```

<div class="image-holder">
  
  
</div>
```

What will be the width of this img?

```
img {
  width: 200px;
  height: 200px;
  border: 1px solid black;
}
```



# CSS precedence

```
<style>
  img{
    width:200px;
    height:200px;
    border:1px solid black;
  }
  img.banner{
    width:500px;
    cursor:pointer;
  }
  .image-holder img.banner{
    width:150px;
  }
</style>
```

```

<div class="image-holder">
  
  
</div>
```

All images will have the same height and a 1 pixel border



# Styling

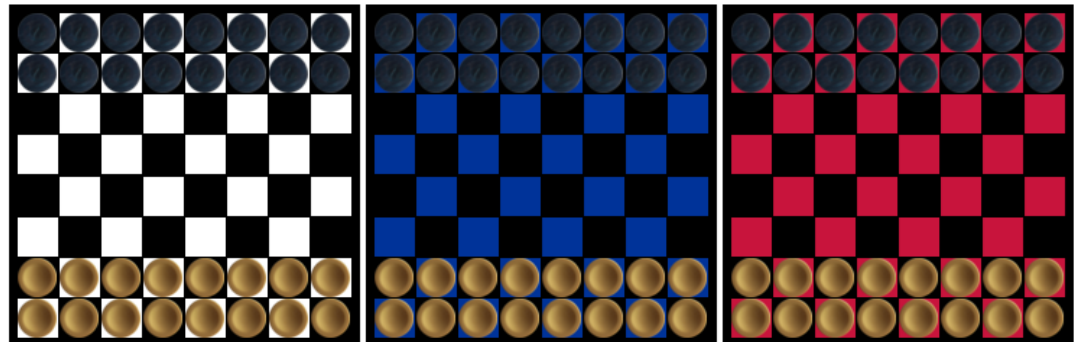
▶ So we need something more specific ...

```
img.banner{
  width: 1192px;
}
```

```
img{
  width:30px;
  height:30px;
  cursor:pointer;
}
```

hello2.html x  
file:///C:/Users/navot/OneDrive/IsraelTechChallenge/becoming%20a%20website%20builder/hello2.html

Welcome To Checkers



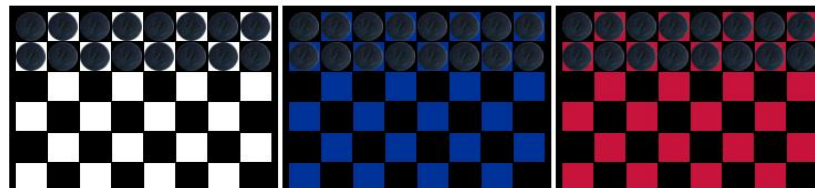
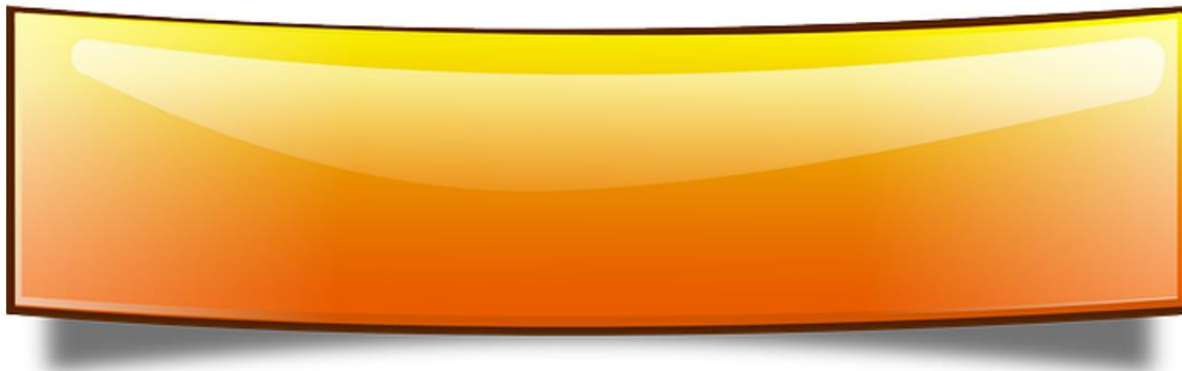
# Questions?



# Styling

- ▶ We want to locate the image at a specific location
- ▶ In order to do that we need to use the position property

Welcome To Checkers



# Position – the General Case

We want to create a box and a button.  
But how will they look like?

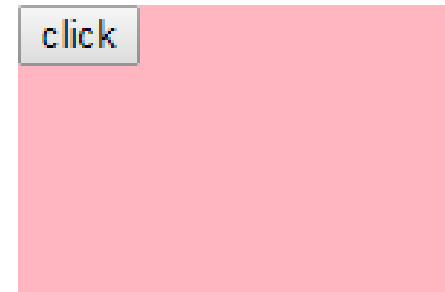
If we have the following html:

```
<body>
  <div class="container ">
    <button>click</button>
  </div>
</body>
```

And the following CSS:

```
<style>
  .container {
    width: 150px;
    height: 100px;
    background: lightpink;
  }
</style>
```

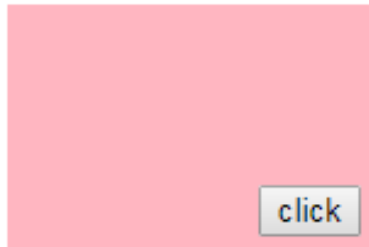
The result will be:



This is the default position.

# Position – the General Case

What if we want the button to be on the bottom right corner?



# Styling

▶ The position style property has 4 different values

static

fixed

Absolute

Relative

# Styling

First we choose the position.

Then, elements can be positioned using the `top`, `bottom`, `left`, and `right` properties

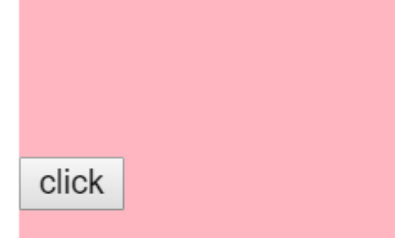
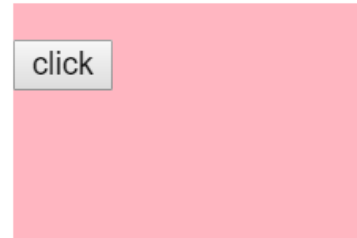
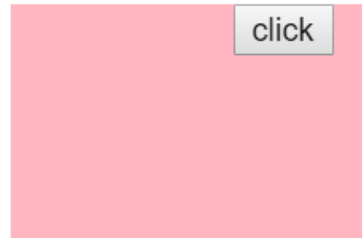
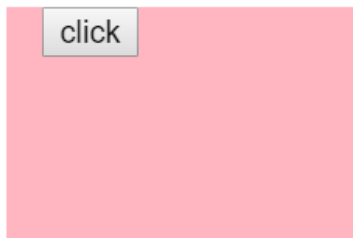
They will be moved away, usually by a few pixels.

`left: 15px;`

`right: 15px;`

`top: 15px;`

`bottom: 15px;`



But away from what?

That will be dictated by the position type!

# Static Position

- ▶ HTML elements are positioned as static by default.
- ▶ Static positioned elements are not affected by the top, bottom, left, and right properties.
- ▶ An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page



# Fixed Position

- ▶ An element with **position: fixed;** is positioned relative to the viewport (the screen)  
what is a viewport? Let's see!
  - it always stays in the same place even if the page is scrolled.
- ▶ The top, right, bottom, and left properties are used to position the element.
- ▶ A fixed element does not leave a gap in the page where it would normally have been located.

# Relative Position

- ▶ An element with **position: relative;** is positioned relative to its normal position (relative to its static position).
- ▶ Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position.
  - Top:20px; will shift it 20 pixels down from the original location. While top: -20px will push it up.
- ▶ The element leaves a gap in the place it was.
- ▶ Other content will not be adjusted to fit into any gap left by the element.

[Live code example](#)

# Absolute Position

- ▶ An element with **position: absolute;** is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
- ▶ However: if an absolute positioned element has no positioned ancestors, it refers to the document body.
  - These elements are not effected by other elements nor do they effect them.
- ▶ Note: A "positioned" element is one whose position is anything except static.

[Live code example](#)

# Positions

## Example

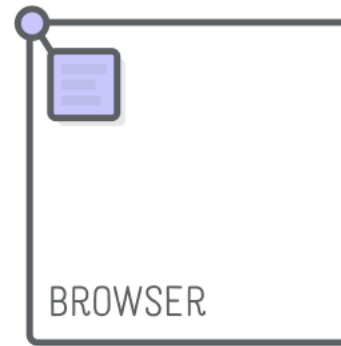
**STATIC**



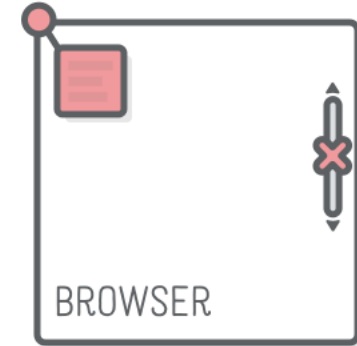
**RELATIVE**



**ABSOLUTE**



**FIXED**



# Positioning

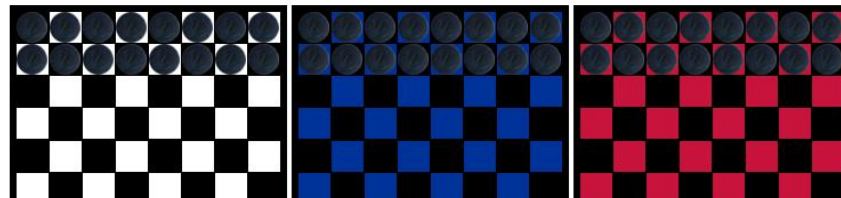
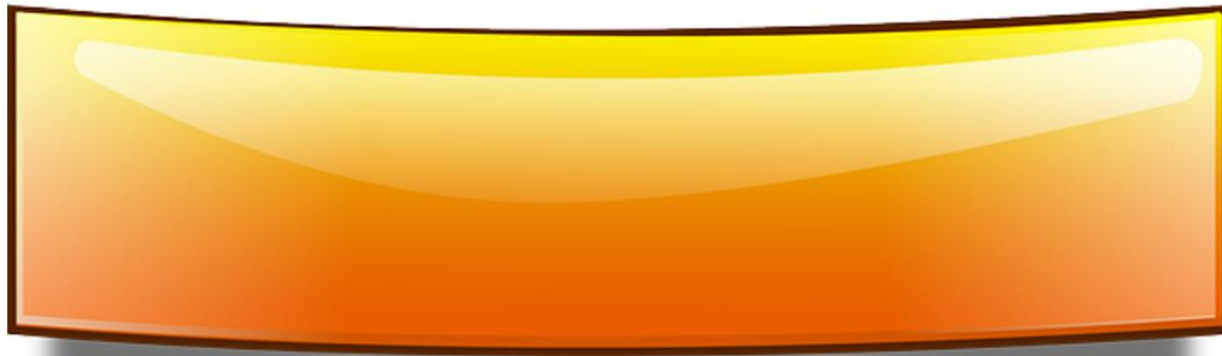
**This is a very complex topic, which is  
very hard to grasp at first.**

**Google is your best friend!**

# So that is where we were

▶ Now we want to locate the image at a specific location

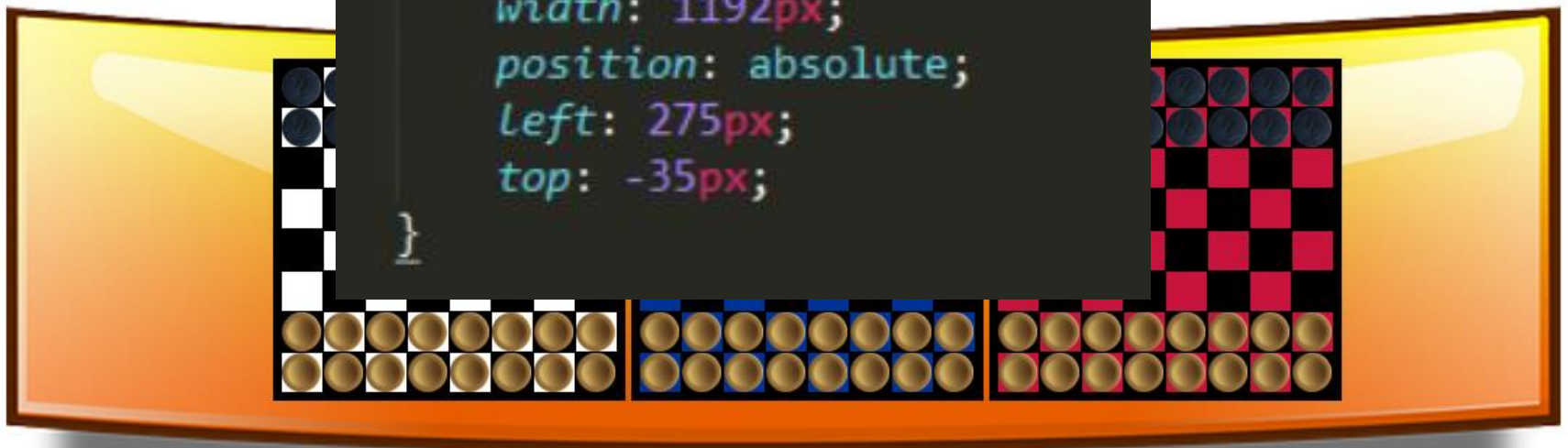
Welcome To Checkers



# Styling

- ▶ What position value will fit our case best?
- ▶ The top and left values where given by the designer: top - 35px, left 275px.

```
img.banner{
  width: 1192px;
  position: absolute;
  left: 275px;
  top: -35px;
}
```



# Styling

- ▶ This is what happened
- ▶ The button is positioned perfectly but it is hiding the boards

---

**Welcome To Checkers**





# Styling

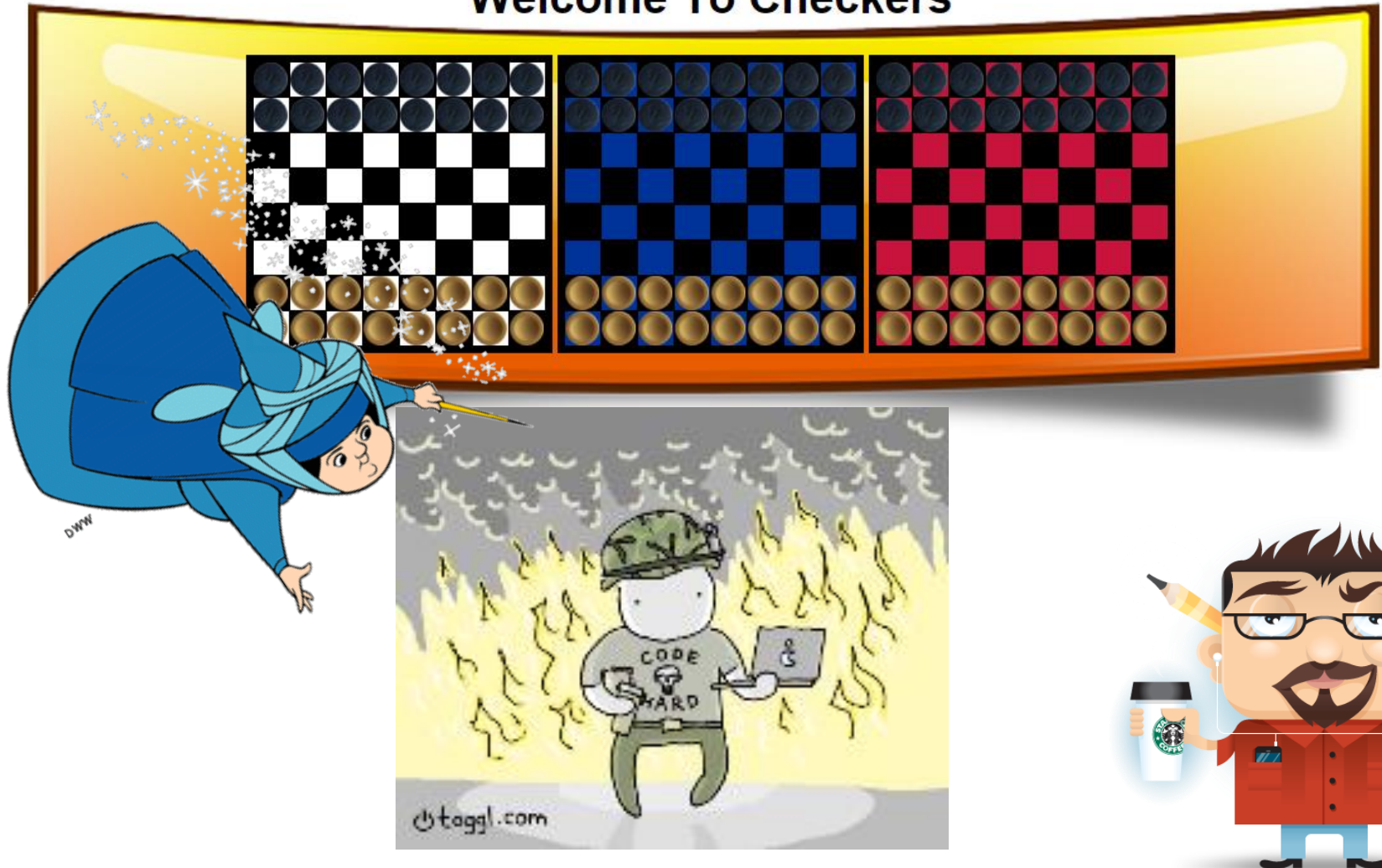
- ▶ We will use our last concept for today
- ▶ The z-index property specifies the stack order of an element.
- ▶ An element with greater stack order is always in front of an element with a lower stack order.
- ▶ Note: z-index only works on positioned elements like
- ▶ (position:absolute, position:relative, or position:fixed).

```
.boards-holder{
  text-align:center;
  z-index:2;
  position:relative;
}

img.banner{
  width: 1192px;
  position: absolute;
  left: 275px;
  top: -35px;
  z-index: 1;
}
```

# Styling

## Welcome To Checkers



# Questions?



# Summary

## ▶ You needed to understand:

- Motivation – why do we learn CSS?
- How to work with CSS selectors (by tag, class or combined)
- CSS precedence

## ▶ You need to remember:

- Use the style tag and never write inline style again

## ▶ You need to practice:

- Display property
- CSS position



# Questions



# Cheat Sheet

```
css selectors
Tagname {
  propname:propvalue;
}
❖ class selector
.dark{
  width: 30px;
}
❖ if the selectors are following (no space) they refer to the same element: td.light
❖ descendants selector – noted by space
table.white-table td.light{
  background-color: rgb(255,255,255);
  cursor: pointer;
  text-align: center;
}
```

Position	Relative to	Leaves gap (top, left ...)
Static	Normal position	Not affected
Fixed	View port	X
Relative	It's static position	V
Absolute	Nearest positioned ancestor	X

Display property

**Inline** (span, img, a)

- contain inline elements and content
- can't have width or height
- do not begin on a new line (attached to the preceding element)

**Inline-block**

like inline elements, but you can set their width and height

**Block** (div, h1, p)

- contain block elements, inline elements and content
- When no space limitation, can take up the entire screen
- begin on a new line and take all of that line

**None**

The document is rendered like the element doesn't exist