CSS3 Fluid Grids

grids1.html

Refer Notebook for CSS code

```
<body>
     <div class="wrapper">
       <div>
         Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quod maxime dicta
aperiam nulla sint mollitia, ratione explicabo ea, soluta deserunt minima necessitatibus
ipsa itaque. Totam doloremque voluptatum pariatur, provident ab ut odio magni beatae
quibusdam voluptate minima, quo laboriosam sequi numquam autem facilis deleniti
laudantium nemo cum! Sed cumque perferendis, quis autem ex consectetur, aperiam
quod quaerat veniam odio, libero sunt recusandae non ipsum tempora aut. Asperiores
possimus rerum voluptate, saepe, quod quis sequi similique nostrum vel! Cum
laudantium, optio.
       </div>
       <div>
         Lorem ipsum dolor sit amet, consectetur adipisicing elit. Architecto nisi amet
ullam iusto saepe, autem? Numquam corporis nihil tempore, eius, sit itaque ex
quibusdam dicta saepe eveniet amet dignissimos nesciunt?
       </div>
       <div>
         Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quod maxime dicta
aperiam nulla sint mollitia, ratione explicabo ea, soluta deserunt minima necessitatibus
ipsa itaque. Totam doloremque voluptatum pariatur, provident ab ut odio magni beatae
quibusdam voluptate minima, quo laboriosam sequi numquam autem facilis deleniti
laudantium nemo cum! Sed cumque perferendis, quis autem ex consectetur, aperiam
quod quaerat veniam odio, libero sunt recusandae non ipsum tempora aut. Asperiores
possimus rerum voluptate, saepe, quod quis sequi similique nostrum vel! Cum
laudantium, optio.
       </div>
       <div>
         Lorem ipsum dolor sit amet, consectetur adipisicing elit. Architecto nisi amet
ullam iusto saepe, autem? Numquam corporis nihil tempore, eius, sit itaque ex
quibusdam dicta saepe eveniet amet dignissimos nesciunt?
       </div>
     </div>
  </body>
</html>
```

grids2.html

```
<html>
  <head>
    <title>CSS Grids</title>
    <style>
      div.wrapper > div:nth-child(even) {
        background-color:#bbb;
        padding: 1em;
      }
      div.wrapper > div:nth-child(odd) {
        background-color:#ddd;
        padding: 1em;
      }
      .wrapper {
        display: grid;
        /* grid-template-columns: 2fr 1fr 2fr 1fr; */
        grid-template-columns: repeat(2,2fr 1fr);
        grid-gap: 1em;
      }

    </style>
  </head>
  <body>
      <div class="wrapper">
        <div>
          Lorem ipsum dolor sit.
        </div>
        <div>
          Lorem ipsum dolor sit amet, consectetur adipisicing elit. Aliquid sint pariatur at
amet assumenda libero molestiae in optio consequuntur ipsum!
        </div>
        <div>
          Lorem ipsum dolor sit.
        </div>
        <div>
          Lorem ipsum dolor sit.
        </div>
        <div>
          Lorem ipsum dolor sit.
        </div>
      </div>
  </body>
</html>
```

grids3.html

```
<html>
  <head>
    <title>CSS Grids</title>
    <style>
      div.wrapper > div:nth-child(even) {
        background-color:#bbb;
        padding: 1em;
      }
      div.wrapper > div:nth-child(odd) {
        background-color:#ddd;
        padding: 1em;
      }
```

Refer Notebook

```
    </style>
  </head>
  <body>
      <div class="wrapper">
        <div class="box" id="box1">Box 1</div>
        <div class="box">Box 2</div>
        <div class="box">Box 3</div>
        <div class="box">Box 4</div>
      </div>
  </body>
</html>
```

grids_responsive.html

```
<html>
  <head>
    <title>Responsive Grids</title>
    <style>
      body{
        background-color: #bbb;
        font-family: verdana;
      }
      .wrapper {
          display: grid;
          grid-template-columns: repeat(4,1fr);
          grid-gap: 1em;
          grid-template-areas:
          "logo header header header"
          "article article article article"
          "i1 i2 i3 i4"
          "p1 p2 p3 p4";
      }
      @media screen and (max-width:600px){
        .wrapper {
          display: grid;
          grid-template-columns: repeat(4,1fr);
          grid-gap: 1em;
          grid-template-areas:
          "logo header header header"
          "article article article article"
          "i1 i1 i2 i2"
          "p1 p1 p2 p2"
          "i3 i3 i4 i4"
          "p3 p3 p4 p4";
        }
      }
      @media screen and (max-width:400px){
        .wrapper {
          display: grid;
          grid-template-columns: repeat(4,1fr);
          grid-gap: 1em;
          grid-template-areas:
          "logo header header header"
          "article article article article"
          "i1 i1 i1 i1"
          "p1 p1 p1 p1"
          "i2 i2 i2 i2"
          "p2 p2 p2 p2"
          "i3 i3 i3 i3"
          "p3 p3 p3 p3"
          "i4 i4 i4 i4"
          "p4 p4 p4 p4";
        }
      }
        #logo{
         grid-area: logo;
        }
        header{
         grid-area: header;
        }
        article{
         grid-area: article;
        }
        #i1{
         grid-area: i1;
        }
        #i2{
         grid-area: i2;
        }
        #i3{
         grid-area: i3;
        }
        #i4{
         grid-area: i4;
        }
        #p1{
         grid-area: p1;
        }
        #p2{
         grid-area: p2;
        }
        #p3{
         grid-area: p3;
        }
        #p4{
         grid-area: p4;
        }
        img{
         max-width:100%;
        }
        p, article{
         background: #ddd;
         padding: 1em;
        }

       </style>
      </head>
```

```
  <body>
      <div class="wrapper">
        <img src="images/logo1.png" id="logo">
        <header>
          <h1>Welcome to Study Circle - Circle of Excellence</h1>
        </header>
        <article>
          Study Circle was founded by Prof. Sandeep Gupta in January 2008 with an aim to
provide superior quality tutoring to degree engineering students. Study Circle believes in
two basic principles - transparency and commitment. It is for this reason that more than
1000 students prefer being a part of this circle every year.
          Prof. Sandeep Gupta (often referred as Sandeep Sir) has been responsible in
perfecting programming skills of thousands of students all over Mumbai. He is renowned
for his expertise in programming subjects like C Programming (SPA), Java Programming
(OOPM), Data Structures, Analysis of Algorithms & Oracle Java Certification (OCJP /
SCJP).
          Apart from providing finest teaching, Study Circle also believes in providing
unyielding infrastructure so that the teaching faculty and students have virtually no
inconvenience. No wonder, our classrooms are much-liked by all our students.
          Summing up, we would say that our endeavour is to excel in everything we do.
And that is why we call Study Circle the "Circle of Excellence."
        </article>
        <img src="images/spa.jpg" id="i1">
        <img src="images/ds.jpg" id="i2">
        <img src="images/oopm.jpg" id="i3">
        <img src="images/ocjp.jpg" id="i4">
        <p id="p1"> SPA is a subject of sem2 and contains the C programming language.
            It is the foundation of programming. </p>
        <p id="p2"> DS & AOA are subjects of sem 3 and  4 respectively and
            very important subjects for placement in dream companies.</p>
        <p id="p3"> OOPM is a subject of sem 3 and contains Java Programming.
            It is the first subject where you will learn java programming. </p>
        <p id="p4"> This is not a subject of MU syllabus. It is an international certification
exam on Java.Many students do this certification to get an edge over others. </p>
      </div>
  </body>
</html>
```

CSS3 Selectors

Overview of CSS 3 selector syntax

Selector type	Pattern	Description
Substring matching attribute selector	E[att^="val"]	Matches any E element whose att attribute value begins with "val".
Substring matching attribute selector	E[att$="val"]	Matches any E element whose att attribute value ends with "val"
Substring matching attribute selector	E[att*="val"]	Matches any E element whose att attribute value contains the substring "val".
Structural pseudo-class	E:root	Matches the document's root element. In HTML, the root element is always the HTML element.
Structural pseudo-class	E:nth-child(n)	Matches any E element that is the n-th child of its parent.
Structural pseudo-class	E:nth-last-child(n)	Matches any E element that is the n-th child of its parent, counting from the last child.
Structural pseudo-class	E:nth-of-type(n)	Matches any E element that is the n-th sibling of its type.
Structural pseudo-class	E:nth-last-of-type(n)	Matches any E element that is the n-th sibling of its type, counting from the last sibling.
Structural pseudo-class	E:last-child	Matches any E element that is the last child of its parent.
Structural pseudo-class	E:first-of-type	Matches any E element that is the first sibling of its type.
Structural pseudo-class	E:last-of-type	Matches any E element that is the last sibling of its type.
Structural pseudo-class	E:only-child	Matches any E element that is the only child of its parent.
Structural pseudo-class	E:only-of-type	Matches any E element that is the only sibling of its type.
Structural pseudo-class	E:empty	Matches any E element that has no children (including text nodes).
Target pseudo-class	E:target	Matches an E element that is the target of the referring URL.
UI element states pseudo-class	E:enabled	Matches any user interface element (form control) E that is enabled.
UI element states pseudo-class	E:disabled	Matches any user interface element (form control) E that is disabled.
UI element states pseudo-class	E:checked	Matches any user interface element (form control) E that is checked.
UI element fragments pseudo-element	E:selection	Matches the portion of an element E that is currently selected or highlighted by the user.
Negation pseudo-class	E:not(s)	Matches any E element that does not match the simple selector s.
General sibling combinator	E~F	Matches any F element that is preceded by an E element.

A) Substring matching attribute selectors

This whole group of selectors is new, and the selectors in it let developers match substrings in the value of an attribute.

Assume that you have an HTML document that contains the following markup:

```
<div id="nav-primary"></div>
<div id="content-primary"></div>
<div id="content-secondary"></div>
<div id="tertiary-content"></div>
<div id="nav-secondary"></div>
```

By using the substring matching attribute selectors you can target combinations of these structural parts of the document.

The following rule will set the background colour of all div elements whose id begins with "nav":

```
div[id^="nav"] { background:#ff0; }
```
In this case the selector will match div#nav-primary and div#nav-secondary.

To target the div elements whose id ends with "primary", you could use the following rule:

```
div[id$="primary"] { background:#ff0; }
```
This time the selector will match div#nav-primary and div#content-primary.

The following rule will apply to all div elements whose id contain the substring "content":

```
div[id*="content"] { background:#ff0; }
```
The elements that will be affected by this rule are div#content-primary, div#content-secondary, and div#tertiary-content.

B) Structural pseudo-classes

The structural pseudo-classes allow developers to target elements based on information that is available in the document tree but cannot be matched by other simple selectors or combinators. The structural pseudo-classes are very powerful, but unfortunately current browsers support only a few of them.

The :root pseudo-class

The :root pseudo-class targets the document's root element. In HTML, the root element is always the HTML element, which means that the following two rules are the same (well, almost - :root has a higher specificity than html):

```
:root { background:#ff0; }
html { background:#ff0; }
```

The :root pseudo-class is currently supported by browsers based on Mozilla and Safari.

The :nth-child() pseudo-class

The :nth-child() pseudo-class targets an element that has a certain number of siblings before it in the document tree. This argument, which is placed within the parentheses, can be a number, a keyword, or a formula.

A number b matches the b-th child. The following rule applies to all p elements that are the third child of their parent element:

```
p:nth-child(3) { color:#f00; }
```

The keywords odd and even can be used to match child elements whose index is odd or even. The index of an element's first child is 1, so this rule will match any p element that is the first, third, fifth, and so on, child of its parent element:

```
p:nth-child(odd) { color:#f00; }
```

The following rule matches p elements that are the second, fourth, sixth, and so on, child of their parent element:

```
p:nth-child(even) { color:#f00; }
```

The formula an + b can be used to create more complex repeating patterns. In the formula, a represents a cycle size, n is a counter starting at 0, and b represents an offset value. All values are integers. Understanding how this works is easier when you look at a few code examples, so let's do that.

The following rules will match all p elements whose index is a multiple of 3. In the first rule, b is zero and could have been omitted, as in the second rule:

```
p:nth-child(3n+0) { color:#f00; }
p:nth-child(3n) { color:#f00; }
```

The :nth-last-child() pseudo-class

The :nth-last-child() pseudo-class works just like the :nth-child() pseudo-class, except that it targets an element that has a certain number of siblings after it in the document tree. In other words, it starts counting from the last child instead of the first, and counts backwards. The following rule will match the second-to-last tr element of a table:

```
tr:nth-last-child(2) { background:#ff0; }
```
The :nth-last-child() pseudo-class is currently not supported by any browsers.

The :nth-of-type() pseudo-class

The :nth-of-type() pseudo-class works exactly like the :nth-child() pseudo-class, but only counts those elements that are of the same type as the element the rule is applied to. This rule will match every p element that is the third p element of its parent:

```
p:nth-of-type(3) { background:#ff0; }
```
This can be useful if you want to make sure that you are really targeting the third p element. At first you might think that you could just as well use the nth-child pseudo-class, but :nth-child(3) will take all sibling elements into account, so the result will be different unless all p elements only have siblings that are also p elements.

The :nth-of-type() pseudo-class is currently not supported by any browsers.

The :nth-last-of-type() pseudo-class

The :nth-last-of-type() pseudo-class targets an element that has a certain number of siblings of the same element type after it in the document tree. Just like the :nth-last-child() pseudo-class, it starts counting from the last child instead of the first, and counts backwards. The following rule will match each second-to-last sibling of type p:

```
p:nth-last-of-type(2) { background:#ff0; }
```
The :nth-last-of-type() pseudo-class is currently not supported by any browsers.

The :last-child pseudo-class

The :last-child pseudo-class targets an element that is the last child of its parent element. It is the same as :nth-last-child(1). This rule will match all p elements that are the last child of their parent element:

```
p:last-child { background:#ff0; }
```

The :first-of-type pseudo-class

The :first-of-type pseudo-class targets an element that is the first sibling of its type. it is the same as :nth-of-type(1).

```
p:first-of-type { background:#ff0; }
```
The :first-of-type pseudo-class is currently not supported by any browsers.

The :last-of-type pseudo-class

The :last-of-type pseudo-class targets an element that is the last sibling of its type. it is the same as :nth-last-of-type(1).

```
p:last-of-type { background:#ff0; }
```

The :last-of-type pseudo-class is currently not supported by any browsers.

The :only-child pseudo-class

The :only-child pseudo-class targets an element whose parent element has no other element children. It is the same (but with a lower specificity) as :first-child:last-child or :nth-child(1):nth-last-child(1).

```
p:only-child { background:#ff0; }
```

The :only-child pseudo-class works in browsers based on Mozilla. Safari seems to interpret it as :first-child (the above rule matches all p elements in the document that are the first child of their parent element).

The :only-of-type pseudo-class

The :only-of-type pseudo-class targets an element whose parent element has no other children of the same element type. It is the same (but with a lower specificity) as :first-of-type:last-of-type or :nth-of-type(1):nth-last-of-type(1).

```
p:only-of-type { background:#ff0; }
```

The :only-of-type pseudo-class is currently not supported by any browsers.

The :empty pseudo-class

The :empty pseudo-class targets an element that has no children. That includes text nodes, so of the following elements, only the first is empty:

```
<p></p>
<p>Text</p>
<p><em></em></p>
```

The following CSS rule will match that first element:

```
p:empty { background:#ff0; }
```

The :empty pseudo-class is currently supported by browsers based on Mozilla. Safari erroneously applies the rule to all elements of the given element type.

The negation pseudo-class

The negation pseudo-class, written :not(s), takes a simple selector as an argument. It targets elements that are not targeted by the simple selector. For example, the following CSS will target any element that is not a p element:

```
:not(p) { border:1px solid #ccc; }
```

The negation pseudo-class currently works in browsers based on Mozilla and Safari.

C) UI element states pseudo-classes

The :enabled and :disabled pseudo-classes

The :enabled and :disabled pseudo-classes allow developers to specify the appearance of user interface elements (form controls) that are enabled or disabled, provided that the browser allows styling of form controls. The following rules will apply different background colours to single line text inputs depending on whether they are enabled or disabled:

```
input[type="text"]:enabled { background:#ffc; }
input[type="text"]:disabled { background:#ddd; }
```

The :checked pseudo-class

The :checked pseudo-class allows developers to specify the appearance of checked radio and checkbox elements. Again, this is provided that the browser allows styling of form controls. This CSS rule will apply a green border to checked radio and checkbox elements:

```
input:checked { border:1px solid #090; }
```

D) The ::selection pseudo-element

The ::selection pseudo-element matches the portion of an element that is currently selected or highlighted by the user. One possible use for this would be to control the appearance of selected text.

Only a few CSS properties apply to ::selection pseudo-elements: color, background, cursor, and outline.

The following rule will make the foreground colour of a selection red:

```
::selection { color:#f00;  }
```

E) The negation pseudo-class

The negation pseudo-class, written :not(s), takes a simple selector as an argument. It targets elements that are not targeted by the simple selector. For example, the following CSS will target any element that is not a p element:

```
:not(p) { border:1px solid #ccc; }
```

F) The General sibling combinator

The general sibling combinator consists of two simple selectors separated by a "tilde" (~) character. It matches occurrences of elements matched by the second simple selector that are preceded by an element matched by the first simple selector. Both elements must have the same parent, but the second element does not have to be immediately preceded by the first element. This CSS rule will target ul elements that are preceded by a p element with the same parent:

```
p ~ ul { background:#ff0;        }
```

CSS3 Colors

CSS3 has Supported additional color properties as follows −

- RGBA colors
- HSL colors
- HSLA colors
- Opacity
-

RGBA stands for Red Green Blue Alpha. It is an extension of CSS2. Alpha specifies the opacity of a color and parameter number is a numerical between 0.0 to 1.0. A Sample syntax of RGBA as shown below −

```
#d1 {background-color: rgba(255, 0, 0, 0.5);}
#d2 {background-color: rgba(0, 255, 0, 0.5);}
#d3 {background-color: rgba(0, 0, 255, 0.5);}
```

HSL stands for hue, saturation, lightness. Here Huge is a degree on the color wheel, saturation and lightness are percentage values between 0 to 100%. A Sample syntax of HSL as shown below −

```
#g1 {background-color: hsl(120, 100%, 50%);}
#g2 {background-color: hsl(120, 100%, 75%);}
#g3 {background-color: hsl(120, 100%, 25%);}
```

HSLA stands for hue, saturation, lightness and alpha. Alpha value specifies the opacity as shown in RGBA. A Sample syntax of HSLA as shown below −

```
#g1 {background-color: hsla(120, 100%, 50%, 0.3);}
#g2 {background-color: hsla(120, 100%, 75%, 0.3);}
#g3 {background-color: hsla(120, 100%, 25%, 0.3);}
```

The following example shows opacity property – css3_colors1.html

```
<html>
  <head>
    <style>
      #p1 {background-color:rgba(255,0,0,0.3);}
      #p2 {background-color:rgba(0,255,0,0.3);}
      #p3 {background-color:rgba(0,0,255,0.3);}
    </style>
  </head>
  <body>
    <p>RGBA colors:</p>
    <p id="p1">Red</p>
    <p id="p2">Green</p>
    <p id="p3">Blue</p>
  </body>
</html>
```

The following example shows HSL color property – css3_colors2.html

```
<html>
  <head>
    <style>
      #g1 {background-color:hsl(120, 100%, 50%);}
      #g2 {background-color:hsl(120,100%,75%);}
      #g3 {background-color:hsl(120,100%,25%);}
    </style>
  </head>
  <body>
    <p>HSL colors:</p>
    <p id="g1">Green</p>
    <p id="g2">Normal Green</p>
    <p id="g3">Dark Green</p>
  </body>
</html>
```

The following example shows HSLA color property – css3_colors3.html

```
<html>
  <head>
    <style>
      #d1 {background-color:hsla(120,100%,50%,0.3);}
      #d2 {background-color:hsla(120,100%,75%,0.3);}
      #d3 {background-color:hsla(120,100%,25%,0.3);}
    </style>
  </head>
  <body>
    <p>HSLA colors:</p>
    <p id="d1">Less opacity green</p>
    <p id="d2">Green</p>
    <p id="d3">Green</p>
  </body>
</html>
```

CSS Background Gradients

Linear Gradients

```
<!DOCTYPE html>
<html>
  <head>
    <title>Background Gradients</title>
    <style>
      body{
        font-size: 40px;
        text-align: center;
        /*background: linear-gradient(90deg, black, white, orange);
        background: linear-gradient(45deg, black, white, orange);
        background: linear-gradient(45deg, black, white, orange); */
        background: linear-gradient(to bottom right, black, white, orange);
      }

    </style>
  </head>
  <body>
      <p>Linear Gradients</p>
      <br><br><br><br><br><br><br><br><br><br><br><br>
      <p>Linear Gradients</p>
  </body>
</html>
```

Radial Gradients

```
<!DOCTYPE html>
<html>
  <head>
    <title>Background Gradients</title>
    <style>
      body{
        font-size: 40px;
        text-align: center;
        background-image: radial-gradient(black, white, orange); /* ellipse is default
        background-image: radial-gradient(circle, black, white, orange);
        background-image: radial-gradient(ellipse, black, white, orange); */
      }

    </style>
  </head>
  <body>
      <p>Linear Gradients</p>
      <br><br><br><br><br><br><br><br><br><br><br><br>
      <p>Linear Gradients</p>
  </body>
</html>
```

CSS Transitions

CSS Transitions allow changes in CSS property values to occur smoothly over a specified duration.

The properties of a transition

A transition can be declared using up to four properties or a single shorthand declaration including all four:

• **transition-property:** the name of the CSS property to be transitioned (such as background-color, text-shadow, or all to transition every possible property).

• **transition-duration:** the length of time over which the transition should occur (defined in seconds, for example 2s or 1.5s).

• **transition-timing-function:** how the transition changes speed during the duration (for example ease, linear, ease-in, ease-out, ease-in-out, or cubic-bezier).

• **transition-delay:** an optional value to determine a delay before the transition commences. Alternatively, a negative value can be used to commence a transition immediately but part way through its transition 'journey'.

transition-timing-function

The transition-timing-function property can have the following values:

ease - specifies a transition effect with a slow start, then fast, then end slowly (this is default)

linear - specifies a transition effect with the same speed from start to end

ease-in - specifies a transition effect with a slow start

ease-out - specifies a transition effect with a slow end

ease-in-out - specifies a transition effect with a slow start and end

cubic-bezier(n,n,n,n) - lets you define your own values in a cubic-bezier function *transition1.html*

```
<!DOCTYPE html>
<html>
  <head>
    <title>CSS Transitions</title>
    <style>
      h2{
        background: red;
        color: white;
        text-align: center;
        width:150px;
        height:50px;

        transition-property: all;
        transition-duration: 2s;
        transition-timing-function: ease;
        transition-delay: 0s;

        /*
        transition: all 2s ease 0s;
        transition: all 2s;

        ease by default, delay 0s by default*/
      }

      h2:hover{
        width:300px;
        height:100px;
        color: black;
        box-shadow: 5px 10px 5px #aaa;
      }

    </style>
  </head>
        <body>
                <h2>Transitions</h2>
        </body>
</html>
```

transition2.html

Refer Notebook

CSS Transformations

transform2D.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>CSS 2D Transformations</title>
    <style>
      h2{
        background: red;
        color: white;
        text-align: center;
        width:200px;
        height:60px;

        transition: all 2s ease 0s;
      }

      h2:hover{
        transform: scale(2.5);
        transform: translate(400px, 100px);
        transform: rotate(30deg);
        transform: skew(20deg,10deg);
      }

    </style>
  </head>
        <body>
    <br><br><br>
                  <h2>Transformations 2D</h2>
        </body>
</html>
```

transform3D.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>CSS 3D Transformations</title>
    <style>
      h2{
        background: red;
        color: white;
        text-align: center;
        width:200px;
        height:60px;

        transition: all 2s ease 0s;
      }

      h2:hover{
        transform: rotateX(180deg);
        transform: rotateY(180deg);
        transform: rotateZ(180deg);
      }
    </style>
  </head>
        <body>
    <br><br><br>
                  <h2>Transformations 3D</h2>
        </body>
</html>
```

CSS Animations

animation.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>CSS Animations</title>
    <style>
      h2{
        background: red;
        color: white;
        text-align: center;
        width:100px;
        height:100px;
        position:relative;            /*reqd for motion*/
        animation-name: abc;
        animation-duration:10s;
        /* animation-direction: reverse;
        normal, alternate, alternate-reverse */
      }


      @keyframes abc {
        0%   {background-color:red;}
      25%  {background-color:black;}
      50%  {background-color:yellow;}
      75%  {background-color:green;}
      100% {background-color:red;}
    }
      /*
      @keyframes abc {
        0%   {background-color:red; left:0px; top:0px;}
      25%  {background-color:black; left:200px; top:0px;}
      50%  {background-color:yellow; left:200px; top:200px;}
      75%  {background-color:green; left:0px; top:200px;}
      100% {background-color:red; left:0px; top:0px;}
    }*/

   </style>
  </head>
  <body>
    <br><br><br>
    <h2></h2>
  </body>
</html>
```