A clock is a free-running signal with a cycle time.

A clock may be either *high* or *low*, and alternates between the two states.

The length of time the clock is high before changing states is its *high duration*; the *low duration* is defined similarly.

The *cycle time* of a clock is the sum of its high duration and its low duration.

The *frequency* of the clock is the reciprocal of the cycle time.
State Elements

A state element is a circuit component that is capable of storing a value.

A state element may be either unclocked or clocked.

Clocked state elements are used in synchronous logic
- When should an element that contains state be updated?
- Edge-triggered clocking means that the state changes either on the rising or the falling edge.
- Clock edge acts as a sampling signal that triggers the update of a state element.

A signal that is to be written into a state element must be stable; i.e., it must be unchanging.

If a function is to be computed in one clock cycle, then the clock period must be long enough to allow all the relevant signals to become stable.
A latch is a circuit that has two stable states, and so can store 1 bit of data.

Feedback: output depends both on present inputs and past inputs.

NOR gate output is only 1 when both inputs are 0.

If the output of one NOR gate is 1 then the output of the other must be 0.

Common contemporary terminology is that a latch does not receive a clock signal, and hence is *transparent*. However, usage does vary...
First, let's understand why this is a stable state. Consider the upper NOR gate: it's receiving two 0's, and hence will output a 1. Therefore, the lower NOR gate will receive one 0 and one 1, and hence it will output a 0. And... the current state is stable.

Now, consider what happens if we toggle one of the inputs...
1: Suppose we toggle R to 1
2: Then the upper NOR gate receives a 1 and a 0 and emits 0
3: Then, the lower NOR gate receives two 0's and emits 1
4: Then, the upper NOR gate receives two 1's and continues to emit 0… stability!

So, turning R(set) on toggles the SR latch (from storing 1) to store 0.
And... turning R(reset) off now doesn't change the state of the latch.
Set-reset Latch Details

1: Suppose we toggle S to 1

2: Then the lower NOR gate receives a 1 and a 0 and emits 0

3: Then, the upper NOR gate receives two 0's and emits 1

4: Then, the lower NOR gate receives two 1's and continues to emit 0... stability!

So, turning S(et) on toggles the SR latch (from storing 0) to store 1.

And... turning S(et) off now doesn't change the state of the latch.
The gated D-latch can be derived from the set-reset latch by adding an interface that makes it possible to essentially isolate the set-reset logic:

If the Enable input is 1 then the value of D will immediately be stored by the S-R mechanism.

If the Enable input is 0 then the value of the S-R mechanism is fixed.

The S = R = 1 case cannot occur for the embedded S-R latch, because…
Timing Issues

There is a small, but positive delay between changes in the input values to a logic gate and any resulting change in the gate's output. We call this the *gate delay*.

Consider the following circuit:

![Circuit Diagram]

Logically, the output of the circuit should ALWAYS be 0. Why?

Consider what happens if the input signal A is set to 1:
- A0, A1 and A2 immediately become 1
- after one gate delay, the output X will become 1 since the XOR has inputs of 0 and 1
- at the same time, the output of the AND gate will become 1
- after one more gate delay, the output X will become 0 again

What would happen if the output X were used as input to another circuit?

We can prevent that if we use a clock signal to synchronize operations.
We create a clocked D-latch by connecting the Enable input of the gated D-latch to a clock signal:

The clocked D-latch accepts the input D only when the clock signal is high (1).

However, there is still a hazard… what if the value of D can change more than once during the high-duration of the clock signal?

The clocked D-latch is level-triggered… that is, whether its state can change depends on the level of the clock signal.
Consider what happens when we combine a clocked D-latch and a clocked S-R latch:

The output of the device can only change once per clock cycle… shortly after the clock signal goes low.
Suppose that $D$ is set to 1; nothing happens at all until the clock signal also goes high:

The output of the D-latch goes high (i.e., takes the value of $D$) but only after two gate delays.

By then, the S-R latch is seeing a low clock signal, and so the S-R latch does not change state yet.
Clocked D Flip-flop

Then, when the clock goes low…

The S-R latch sees a high clock signal (after 1 gate delay), and so it updates state.

But, the D-latch sees a low clock signal immediately and so it cannot change state.
The JK flip-flop takes two data inputs and updates its state $Q$, on a clock tick, according to the excitation table:

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
<th>Q</th>
<th>~Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>no change</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>opposite</td>
<td></td>
</tr>
</tbody>
</table>

Commonly, it takes an entire clock cycle for the JK flip-flop to update its state, and so the change in state is commonly seen on the falling edge of the clock cycle.

It is also common to provide additional input connections for clear and reset and enable signals.
4-Bit Register

Built using D flip-flops:

Clock input controls when input is "written" to the individual D flip-flops.

This is easily scaled to store a wider data value.